Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Disseny i construcció d'aparells electrònics per identificar el color d'objectes

MEMÒRIA

Autores: Cristina Rodríguez Murciano
Judit Sanahuja Roig
Directora: Rosa Rodríguez Montañés

Convocatòria: Gener 2017

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

Aquest projecte consisteix en el disseny i la construcció de dos sistemes electrònics, basats en un microcontrolador PIC, que identifiquen el color d'objectes. Els dos sistemes capten el feix de llum reflectit per l'objecte mitjançant un sensor de color, però la diferència entre ells és que en un sistema la informació del color es llegeix en un dispositiu mòbil amb sistema operatiu *Android* i a l'altre, en canvi, en un dispositiu LCD.

Pel que fa al sistema amb el dispositiu mòbil, a part d'identificar el color captat, també és capaç de generar un color de components seleccionades per l'usuari en un LED RGB (format per tres díodes emissors de llum, un de vermell, un de verd i un de blau,en un mateix encapsulat). La comunicació entre el sistema i el dispositiu mòbil es realitza mitjançant tecnologia *Bluetooth*. S'ha desenvolupat una aplicació *Android* per controlar aquest sistema, es tracta d'una interfície perquè qualsevol usuari la pugui utilitzar.

Per altra banda, el segon sistema conté dispositiu LCD, i com en l'altre sistema, també genera un color en un LED RGB, però en aquest cas no un qualsevol sinó que reprodueix el captat pel sensor. En aquest sistema s'introdueix un polsador per controlar quan s'encén el díode.

Per als dos sistemes, la programació del microcontrolador s'ha realitzat amb el llenguatge C i s'ha realitzat un programa per sistema. En cadascun d'ells s'han programat totes les funcions que té el sistema corresponent.

Així doncs, en aquest document es detallen els components que s'han utilitzat per a la realització d'ambdós sistemes, així com els passos que s'han seguit per implementar-los. També s'explica la programació d'aquests, tant la dels microcontroladors com la de l'aplicació *Android* utilitzada pel funcionament del sistema amb el dispositiu mòbil.

Sumari

1. Glossari

***App*:** Aplicació mòbil. És una aplicació informàtica dissenyada en telèfons intel·ligents, tauletes tàctils i altres dispositius mòbils.

***Baud rate*:** Velocitat en bauds. Nombre d'unitats de senyal per segon. Un baud pot contenir varis bits.

Bit de paritat: Dígit binari que indica si el nombre de bits amb valor 1 en un conjunt de bits és parell o senar. Mètode de detecció d'errors de transmissió o emmagatzemament de dades.

CPU (de l'anglès *Central Processing Unit*): Unitat central de processament. Component de dispositius programables que interpreta les instruccions contingudes en els programes i processa les dades.

Díode: Dispositiu electrònic proveït de dos terminals que deixa passar el corrent només en un sol sentit.

EEPROM (de l'anglès *Electrically Erasable Programmable Read-Only Memory*): Tipus de memòria ROM que pot ser programada, esborrada i reprogramada elèctricament.

EUSART (de l'anglès *Enhanced Universal Synchronous Asynchronous Receiver-Transmitter*): Dispositiu que converteix dades de paral·lel a sèrie o de sèrie a paral·lel per tal de transmetre o rebre informació, respectivament.

I/O (de l'anglès *Input/Output*): Entrada/sortida.

LED (de l'anglès *Light Emitting Diode*): Díode emissor de llum.

Microcontrolador: Circuit integrat que inclou una CPU, unitats de memòria, ports d'entrada i sortida i perifèrics.

Microprocessador: Circuit integrat que pot contenir una o més CPUs.

Perifèric: Aparell o dispositiu auxiliar i independent connectat a la CPU.

PIC (de l'anglès *Peripheral Interface Controller*): Família de microcontroladors fabricats per *Microchip Technology Inc*.

***Protoboard*:** Placa de proves usada per construir prototips de circuits electrònics sense la

necessitat de realitzar soldadures.

PWM (de l'anglès *Pulse-Width Modulation*): Modulació per l'amplada de polsos. Tècnica de modulació emprada per a codificar un missatge en una sèrie de polsos d'amplada variable.

RAM (de l'anglès *Random Access Memory*): Memòria d'accés aleatori d'escriptura i lectura.

RGB (de l'anglès *Red, Green, Blue*): És la composició del color en termes de la intensitat dels colors primaris de la llum: el vermell, el verd i el blau.

ROM (de l'anglès *Read Only Memory*): Tipus de memòria electrònica de la qual només es pot llegir la informació que té gravada.

SRAM (de l'anglès *Static Random Access Memory*): La memòria estàtica d'accés aleatori és un tipus de tecnologia RAM basada en semiconductors, capaç de mantenir les dades.

UART (de l'anglès *Universal Asynchronous Receiver-Transmitter*): Nom que comunament rep l'EUSART.

2. Prefaci

2.1. Origen i motivació del projecte

La finalitat d'aquest projecte és la realització del Treball Final de Grau per tal de concloure els estudis en el Grau en Enginyeria en Tecnologies Industrials cursat a l'Escola Tècnica Superior d'Enginyeria Industrial de Barcelona (ETSEIB).

Les autores d'aquest projecte es troben cursant l'últim any del grau. Aquest ofereix una visió multidisciplinària de l'enginyeria i permet conèixer diferents camps científics i tecnològics. Abans de cursar l'assignatura obligatòria d'electrònica que s'imparteix al quart curs, les autores van cursar juntes una assignatura optativa que consistia en la construcció d'una figura lluminosa feta amb LEDs. A ambdues els hi va despertar interès el tema de l'electrònica i, sobretot, la motivació per tal d'aplicar els conceptes inicials i teòrics apresos en l'única assignatura d'electrònica del Grau.

Per altra banda, la coneixença des de fa temps de les autores i la bona experiència a l'hora de treballar en equip, va impulsar-les a plantejar-se fer el Treball Final de Grau conjuntament i aprofundir en els coneixements que ja havien prèviament adquirit. A més, ambdues mostren interès per les noves tecnologies i un projecte en aquest àmbit els permet apropar-s'hi una mica més.

Un altre punt que va fer augmentar la motivació en fer un projecte juntes en l'àmbit de l'electrònica és que realitzar un projecte en parella permet abastir uns objectius més ambiciosos i, com succeeix en aquest cas, permet també comparar més d'una forma de resoldre el projecte plantejat. Ambdues creuen que aquest fet enriqueix molt un projecte.

2.2. Requeriments previs

Tot i no ser necessari ser un gran coneixedor de l'electrònica en totes les seves vessants, si que són necessàries unes certes bases. Aquestes han sigut adquirides de manera teòrica en l'assignatura d'electrònica i, de manera més pràctica, en l'assignatura optativa prèviament mencionada on es va tenir el primer contacte amb alguns dels components utilitzats en aquest projecte com ara els LEDs, el programador, les resistències, la protoboard, el soldador...

Per altra banda, tot i que el llenguatge de programació utilitzat en el treball ha sigut el llenguatge C que cap de les dues no havia fet servir mai, ambdues són coneixedores del

llenguatge *Python*, que és l'utilitzat a l'Escola, fet que ha facilitat molt l'adaptació al nou llenguatge.

3. Introducció

3.1. Objectius del projecte

El projecte consisteix en el disseny i la construcció de dos sistemes electrònics, basats en un microcontrolador PIC (*Peripheral Interface Controller*), capaços de captar un feix de llum visible i mostrar informació sobre la composició RGB (*Red, Green, Blue*) per la pantalla d'un telèfon mòbil, mitjançant una aplicació *Android*, i l'altre, per una pantalla LCD (*Liquid Crystal Display*).

A banda de l'objectiu principal, la construcció dels sistemes electrònics esmentats, també hi ha un seguit d'objectius secundaris, entre els quals està poder aplicar els coneixements d'electrònica apresos durant el Grau en Enginyeria en Tecnologies Industrials. També els d'informàtica, útils per a realitzar el programa del microcontrolador; tot i que la programació del PIC s'ha realitzat amb C i durant el grau només s'havia après a programar amb *Python*, la metodologia és semblant, així com la manera d'estructurar el programa.

Per últim, a més de transmetre la informació d'un color a un altre dispositiu, com el mòbil o la LCD, també es reprodueix el color captat pels dos sistemes en un LED RGB.

3.2. Abast del projecte

En aquest apartat es descriuen les limitacions que s'han tingut a l'hora de realitzar aquest projecte, les quals s'han de tenir en compte en tot moment ja que afecten tant al desenvolupament com al resultat del treball.

Per començar, els components i el material que s'han utilitzat en aquest projecte són els que hi ha disponibles al laboratori d'electrònica de l'ETSEIB. En alguns casos, es tracta de dispositius senzills però que són suficients per a obtenir uns bons resultats en l'àmbit que s'ha treballat.

A més, l'aplicació de mòbil s'ha dissenyat per al sistema operatiu *Android*, ja que permet la creació gratuïta d'aplicacions, a diferencia del sistema operatiu d'*Apple*, l'*iOS*, on la creació d'aplicacions és més restringida perquè està més controlada. L'aplicació s'ha programat amb el programari *MIT App Inventor 2*, el qual ha simplificat el procés ja que permet un mode de programació molt intuïtiu mitjançant l'ús de blocs.

Per altra banda, la comunicació entre el microcontrolador i el mòbil s'ha efectuat mitjançant

Bluetooth, un protocol que, a part d'estar molt estès, està a l'abast dels usuaris.

Finalment, com s'ha mencionat en l'apartat anterior, per a la programació del microcontrolador s'ha utilitzat el llenguatge C per ser el més utilitzat en el microcontrolador usat. A més, es tracta d'un llenguatge útil i força fàcil d'aprendre.

4. Especificacions i arquitectura del sistema

En aquest capítol s'explica quina és l'arquitectura del sistema final i les funcions d'aquest separades per dos grans blocs: el dispositiu mòbil i la LCD.

4.1. Dispositiu mòbil

El dispositiu mòbil actua tant de transmissor com de receptor de dades. Quan aquest actua com a transmissor, el color generat des de l'aplicació mòbil és reproduït amb un LED. Quan el dispositiu actua com a receptor es mostra en la pantalla del telèfon el color captat pel sensor. A continuació, s'explica el procés detallat i els components que hi intervenen en ambdós casos.

4.1.1. Visualització dels components del color (ús com a receptor)

Una targeta de color és situada davant del sensor, que és envoltat per una caixa negra per tal d'evitar interferències de la llum exterior. El sensor capta les freqüències de *red*, *green*, *blue* i *clear* que té la cartolina i les envia al microcontrolador. Aquestes dades són processades mitjançant un programa que, al ser executat, les envia via *Bluetooth* al telèfon. Mitjançant una aplicació, es pot visualitzar el color captat pel sensor per la pantalla del telèfon. La relació entre els elements involucrats en aquest procés es pot veure esquematitzat a la *Figura 4.1*.

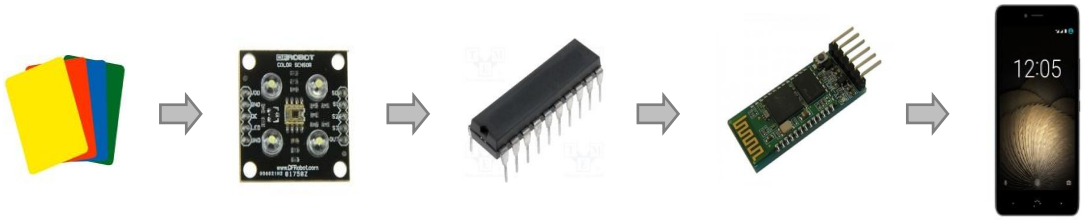
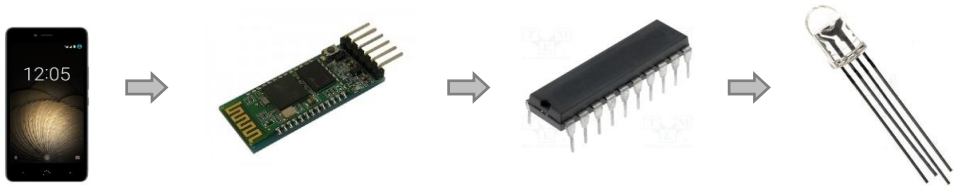


Figura 4.1. Dispositiu mòbil actuant com a receptor de dades

4.1.2. Generació del color sobre un LED (ús com a transmissor)

La mateixa aplicació que permet visualitzar el color per la pantalla, també permet a l'usuari seleccionar els components RGB d'un color i enviar les dades d'aquest via *Bluetooth* al microcontrolador. Aquestes dades són processades mitjançant un programada que, al ser

executat, reprodueix el color generat amb el LED. La relació entre els elements involucrats en aquest procés es pot veure esquematitzat a la *Figura 4.2*.



Dispositiu *Android* | Dispositiu *Bluetooth* HC-05 | Microcontrolador PIC16F690 | LED RGB L-154A4SURKQBDZGW

Figura 4.2. Dispositiu mòbil actuant com a transmissor de dades

4.2. LCD

La pantalla LCD actua com a receptora de la informació obtinguda pel sensor de color. Com en el cas en què el dispositiu mòbil actua de receptor, el sensor capta informació sobre la relació d'intensitat en les diferents freqüències de cada color primari (RGB) de la llum de la targeta situada davant d'aquest dispositiu, enviant els resultats al microcontrolador. Posteriorment, la pantalla mostra la quantitat de vermell, verd, blau i blanc que té el color captat. A més, la informació que rep el microcontrolador, és a dir, les intensitats de les freqüències dels colors, seran emeses per un LED RGB, habilitat per un polsador, de manera que aquest emeti el color que capta el sensor. L'esquema següent (*Figura 4.3)* mostra aquest procés.

Targetes de colors | Sensor color TCS3210 | Microcontrolador PIC16F690 | LCD Hitachi HD44780 controller

Polsador | LED RGB L-154A4SURKQBDZGW

Figura 4.3. Esquema del procés amb la LCD

5. Components

En aquest capítol s'expliquen detalladament cadascun dels components que s'han utilitzat per dur a terme els processos explicats en l'apartat anterior.

En la *Taula 5.1*, es mostra cadascun dels components i els models triats respectivament.

Component	Model
Dispositiu *Android*	BQ U Plus
Bluetooth	HC-05
Microcontrolador	PIC16F690
Sensor de color	TCS3210
LED RGB	L-154A4SURKQBDZGW
LCD	Hitachi HD44780 controller 16x2
Polsador	Normalment desconnectat

Taula 5.1. Models dels components utilitzats en el projecte

El model del dispositiu mòbil triat és el que ja disposaven les autores del projecte abans de començar. Els models dels altres components han estat triats, o bé perquè l'ETSEIB ja els disposava, o bé perquè són fàcilment assequibles, barats i disposen de les característiques necessàries del projecte.

5.1. Microcontrolador PIC16F690

Un microcontrolador és un circuit integrat programable, el qual consta d'un microprocessador, ports d'entrada i sortida, perifèrics i unitats de memòria RAM, memòria d'accés aleatori d'escriptura i lectura, i ROM, memòria de la qual només es pot llegir la informació que té gravada.

En aquest treball s'ha utilitzat el microcontrolador PIC16F690, que és de la família de microcontraladors PIC, de l'anglès *Peripheral Interface Controller*, fabricat per l'empresa

Microchip Technology Inc.

5.1.1. Característiques generals

A continuació, es mostra una taula (*Taula 5.2*), extreta del *Datasheet* del PIC16F690 **[1]**, on es poden observar les característiques generals del microcontrolador usat en aquest treball.

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit	SSP	ECCP+	EUSART
	Flash (words)	SRAM (bytes)	EEPROM (bytes)							
PIC16F690	4096	2566	256	18	12	2	2/1	Yes	Yes	Yes

Taula 5.2. Característiques generals del PIC16F690

Com s'aprecia a la taula, les memòries de dades SRAM, de l'anglès *Static Random Access Memory*, i EEPROM, de l'anglès *Electrically Erasable Programmable Read-Only Memory*, són de 256 bytes. A més, s'observa que, de principals perifèrics, que permeten la comunicació del microcontrolador amb l'exterior, té dos comparadors, tres *timers*, dels quals dos són de 8 bits i un de 16 bits, 18 pins d'entrada i sortida i mòdul EUSART.

El PIC16F690 té un ampli rang de tensió de funcionament, des dels 2,0V fins als 5,5V. També consta de rellotge intern/extern, que executa cada línia d'instrucció per cada 4 senyals de rellotge, on el rellotge intern es pot configurar mitjançant uns bits amb una velocitat que va des dels 32kHz als 8MHz, tot i que per defecte són 4MHz (aquesta és la freqüència utilitzada en aquest projecte).

5.1.2. Diagrama de blocs

En el diagrama de blocs del PIC16F690 (*Figura 5.1*), es poden apreciar els diferents mòduls que té el microcontrolador i el seu interconnexionat. A continuació s'explicaran els mòduls que s'han utilitzat en aquest projecte, ja que no s'han fet servir tots.

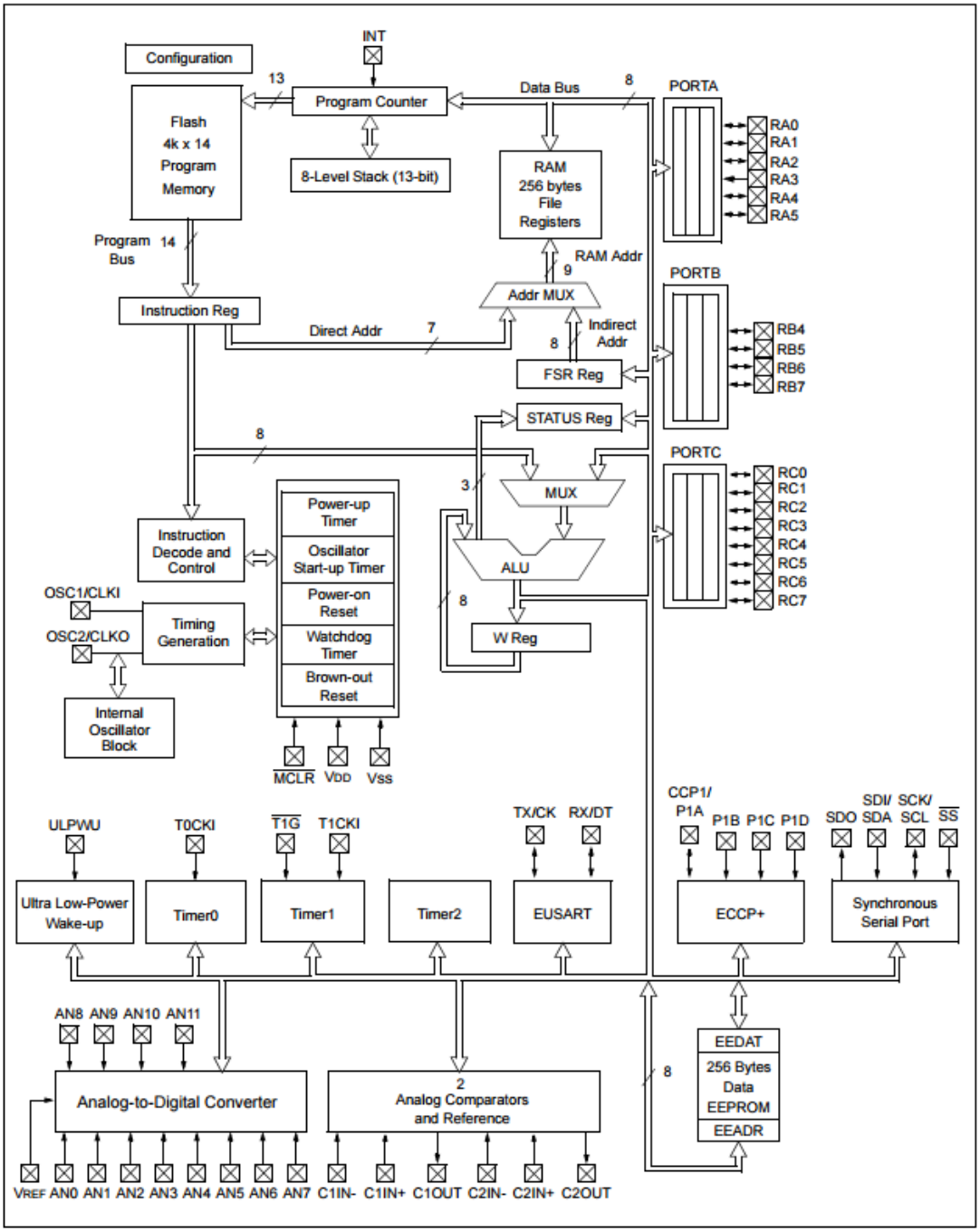


Figura 5.1. Diagrama de blocs del PIC16F690

5.1.3. Memòria

El microcontrolador PIC16F690 té una memòria de dades RAM, de l'anglès *Random Access Memory*. Una memòria RAM, al contrari que una memòria ROM, de l'anglès *Read Only Memory*, permet tant la lectura com l'escriptura de dades i es caracteritza per poder accedir a qualsevol byte sense haver d'accedir als bytes que el precedeixen i, per tant, no hi

ha quasi diferència de temps en accedir a un bit o a un altre.

Una memòria RAM pot ser estàtica, anomenada SRAM (de l'anglès *Static RAM)*, o dinàmica, anomenada DRAM (de l'angles D*ynamic RAM*). La diferència entre una i l'altra és que les DRAM necessiten actualitzar-se milers de vegades per segon i les SRAM no, però aquesta última és més cara.

La RAM del PIC16F690 és de 256 bytes i de tipus SRAM volàtil. Que sigui volàtil implica que pot emmagatzemar informació durant un període infinit de temps mentre estigui alimentada. Si se li treu l'alimentació la informació es perd.

Per tal d'accedir a la memòria RAM, s'utilitza una memòria programable no volàtil ROM, on s'emmagatzema el programa. Aquesta, al ser no volàtil, no perd informació al ser desconnectada de l'alimentació i és reprogramable. El fet que sigui reprogramable permet a l'usuari esborrar o modificar el contingut de la memòria elèctricament. La memòria ROM del PIC utilitzat és de tipus Flash de 4K i s'accedeix mitjançant un bus de dades.

A més, la memòria del PIC16F690 també consta d'una memòria EEPROM, que és un tipus de memòria ROM programable i esborrable elèctricament. La seva funció és que es puguin emmagatzemar dades durant l'execució.

5.1.4. Processador

El microcontrolador d'aquest projecte té un processador RISC, de l'anglès *Reduced Instruction Set Computer)* amb una arquitectura de busos *Harvard*.

L'arquitectura *Harvard* té l'Unitat de Procés Central, CPU (de l'anglès *Central Processing Unit*), connectada a dues memòries: la memòria de programa (emmagatzema només les instruccions) i la memòria de dades (emmagatzema només les dades). Per accedir-hi, la CPU utilitza dos busos diferents independents entre ells. D'aquesta manera, s'hi pot accedir simultàniament i de manera independent a ambdues memòries. Aquesta arquitectura, al contrari que la de *Von Neumann* que només utilitza un bus, accedeix a la memòria de programa mitjançant un bus de dades de 14 bits i a la memòria de dades amb un bus independent de 8 bits. A més, a l'utilitzar dos busos, mentre es realitza una instrucció ja s'està llegint la següent i això li proporciona una velocitat d'execució més alta comparada amb la d'altres arquitectures.

Amb el processador RISC i amb aquest microcontrolador en concret, té un conjunt de 33 instruccions. Per tal de sincronitzar el sistema, s'utilitza el rellotge intern de 4MHz, i cada instrucció necessita quatre cicles de rellotge per executar-se (1µs).

5.1.5. Ports d'entrada i sortida

Com tots els microcontroladors, el PIC16F690 consta de ports d'entrada i sortida, I/O,amb diferents propòsits, amb el principal objectiu de comunicar-se amb l'exterior rebent o enviant dades a través dels seus pins.

Aquest microcontrolador consta de 20 pins, dividits en tres ports paral·lels d'entrada i sortida, el PORTA, el PORTB i el PORTC, de 6, 4 i 8 pins, respectivament, a part del pin d'alimentació, V_{DD}, i el de terra, V_{SS}. Aquests ports I/O poden ser d'entrada, quan el PIC llegeix dades de l'exterior, de sortida, quan, enlloc de rebre, envia dades a l'exterior, o bidireccionals, quan la direcció de les dades, si s'envien o es reben, es pot determinar durant l'execució del programa.

Per altra banda, tot i que generalment els pins s'emprin de forma digital, és a dir, que reben o envien un 0 o un 1, el PIC16F690 permet que 12 dels seus pins s'utilitzin de manera analògica, determinant el seu mode de funcionament a la configuració del programa, per tal de què puguin comunicar-se amb perifèrics analògics.

En la *Figura 5.2*, extreta del *Datasheet* del microcontrolador utilitzat, s'observa com cada pin pot realitzar més d'una funció, tot i que no es poden implementar de forma simultània, així doncs, en la configuració del dispositiu, s'ha d'indicar quina funció és la que es vol executar.

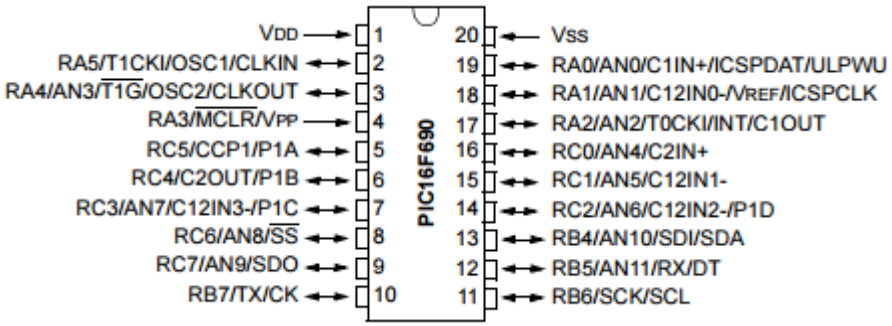


Figura 5.2. Diagrama de pins del PIC16F690

5.1.6. UART

La connexió entre el dispositiu mòbil i el microcontrolador s'ha fet via Bluetooth utilitzant l'EUSART (de l'anglès *Enhanced Universal Synchronous Asyncronous Reciever-Transmitter*) o més comunament conegut com a UART (de l'anglès *Universal Asynchronous Receiver-Transmitter).* Es tracta d'un perifèric d'entrada/sortida, I/O (de l'anglès input/output). La funció principal d'aquest dispositiu és convertir dades de paral·lel a

sèrie o de sèrie a paral·lel per tal de transmetre o rebre informació, respectivament.

Les dues entrades pertinents a aquestes dues funcions són TX per transmetre i RX per rebre i són, respectivament, els pins 10 i 12 del microcontrolador. En aquest cas s'ha utilitzat l'UART tant com a receptor com a transmissor. L'ús com a transmissor s'ha fet servir per a transmetre les dades al telèfon mòbil i poder visualitzar el color captat pel sensor i la seva composició RGB i l'ús com a receptor, per a captar les ordres fetes per l'usuari des de l'aplicació Android.

En l'UART la informació es transmet en grups de 8 bits, és a dir, un byte. El format de la trama de l'UART consisteix en un bit d'inici, una paraula de 8 bits, un bit opcional de paritat i un bit de parada.

En estat de repòs el senyal està a nivell alt, és a dir, a 5V. El bit d'inici connecta el senyal a nivell baix, provocant un flanc de baixada (pas de nivell alt a nivell baix), que indica que a continuació comença la transmissió d'una paraula de 8 bits. Tot seguit, es transmeten els 8 bits de la paraula i, opcionalment, aquesta seqüència ve seguida d'un bit de paritat, que indica si hi ha d'haver un número parell o senar d'1's a la paraula transmesa. Aquest últim bit serveix per detectar errors i en aquest projecte no s'ha cregut necessari fer-lo servir. Finalment, hi ha el bit de parada, on el senyal torna a estar a nivell alt durant un cert període de temps per indicar que la transmissió ha finalitzat. A la *Figura 5.3* es pot veure un esquema de la trama de l'UART, on St (Start bit) és el bit d'inici, del 0 al 8 són els bits que conformen la paraula (Data bits), [P] (Parity bits) és el bit de paritat, Sp (Stop bit) és el bit de parada i en IDLE el senyal està en repòs (no hi ha ni transferència ni recepció de dades).

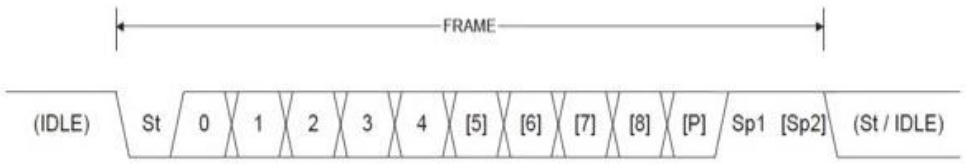


Figura 5.3. Esquema trama UART

L'EUSART és un receptor-transmissor asíncron. Per tant, caldrà fixar una velocitat de transmissió, en anglès anomenada *baud rate*. Tal com s'ha esmentat anteriorment, l'inici de la transmissió es produeix quan hi ha un flanc de baixada. En aquest projecte s'ha fixat una velocitat de transmissió de 9600 bauds, mesura molt comunament utilitzada i suficient per a l'envergadura d'aquest projecte. Per aquesta velocitat, es té un període de 104,2μs.

El PIC16F690 consta d'un propi generador de bauds. Per caracteritzar la velocitat de transmissió desitjada, caldrà inicialitzar alguns registres de l'UART. Per defecte, el *timer*

que s'utilitza és de 8 bits. El senyal de rellotge utilitzat és de 4MHz. Amb aquestes característiques, segons el *Datasheet*, SPBRG ha de prendre el valor de 25 (valor decimal) (*Taula 5.3*).

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)
300	—	—	—	—	—	—	—	—	—	300	0.16	207
1200	1202	0.16	207	1200	0.00	191	1202	0.16	103	1202	0.16	51
2400	2404	0.16	103	2400	0.00	95	2404	0.16	51	2404	0.16	25
9600	9615	0.16	25	9600	0.00	23	9615	0.16	12	—	—	—
10417	10417	0.00	23	10473	0.53	21	10417	0.00	11	10417	0.00	5
19.2k	19.23k	0.16	12	19.2k	0.00	11	—	—	—	—	—	—
57.6k	—	—	—	57.60k	0.00	3	—	—	—	—	—	—
115.2k	—	—	—	115.2k	0.00	1	—	—	—	—	—	—

Taula 5.3. Baud rates per mode asíncrons

Si es volgués utilitzar un *timer* de 16 bits, s'hauria de posar el registre BRG16 a 1 i consultar en la taula adequada quin hauria de ser el valor de SPBRG.

A la *Taula 5.4* es mostren vàries fórmules que permeten calcular el *baud rate* on n és el valor en decimal del registre SPBRG i Fosc la freqüència del senyal del rellotge. A continuació, es comprova que amb els valors de freqüència, SPBRG i *baud rate* escollits es té un marge d'error admissible.

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	Fosc/[64 (n+1)]
0	0	1	8-bit/Asynchronous	Fosc/[16 (n+1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	Fosc/[4 (n+1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Taula 5.4. Fórmules per calcular el baud rate

$$Baud\ rate = \frac{F_{osc}}{16(n+1)} = \frac{4 \cdot 10^6}{16(25+1)} = 9615 \text{ bauds}$$

$$error_{rel} = \frac{9615-9600}{9600} \cdot 100 = 0{,}16\ \%$$

Com que l'error relatiu és d'un 0,16%, es considera vàlid l'ús d'aquests paràmetres.

A la *Taula 5.5*, es troben els registres associats al generador de bauds, els quals s'han utilitzat per inicialitzar l'UART.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BAUDCTL	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D

Taula 5.5. Registres associats al generador de bauds

En el fitxer uart.h (veure *Annex 1.1*), a la funció UART_Init, s'inicialitzen els registres TXSTA, SPBRG, BRG16, RCSTA. Al capítol 8 s'explica amb detall quins valors prenen els bits d'aquests registres.

L'UART pot tant rebre com transmetre. A continuació, s'expliquen els dos modes de treball.

En la *Figura 5.4*, s'observa un esquema de l'UART com a receptor. Les dades entren per RX i es transmeten bit a bit al *RSR Register.* Quan aquest està ple, la paraula es transmet al *RCREG Register* i es produeix una interrupció. Quan es produeix una interrupció, el programa deixa d'executar la funció principal i executa una funció específica d'interrupció (ISR, de l'anglès *Interrupt Service Routine*) que tracta la situació en qüestió.

El *Baud Rate Generator*, genera la velocitat de transmissió necessària caracteritzada prèviament en la inicialització de l'UART.

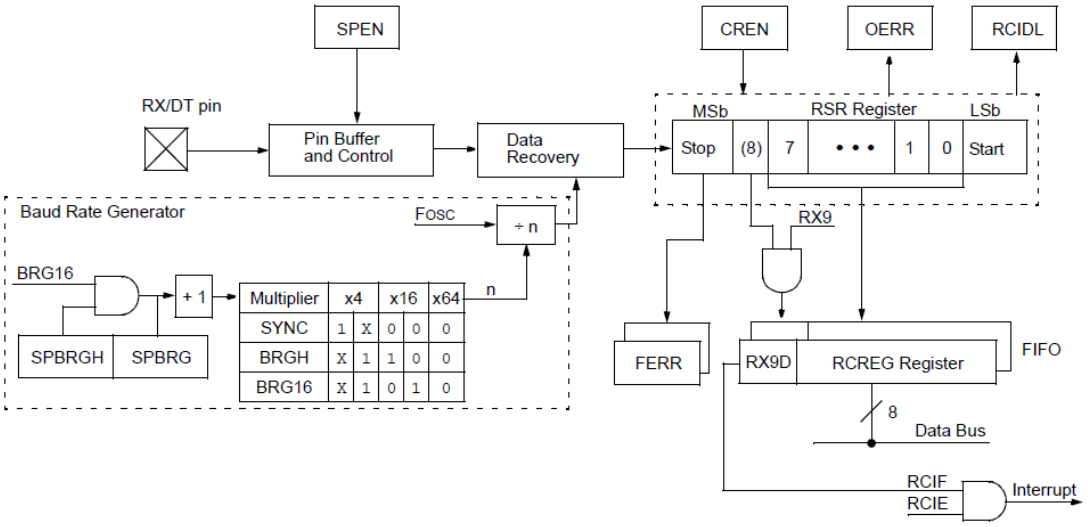


Figura 5.4. Esquema UART com a receptor

En la *Figura 5.5*, l'UART té la funció de transmetre. La paraula d'un byte que es vol transmetre s'escriu al *TXREG Register* i el propi hardware s'encarrega de copiar-ho al *Transmit Shift Register,* i d'allà a TX. Igual que en el cas anterior, s'executa una interrupció.

Anàlogament, en el Baud Rate Generator, es genera la velocitat de transmissió desitjada.

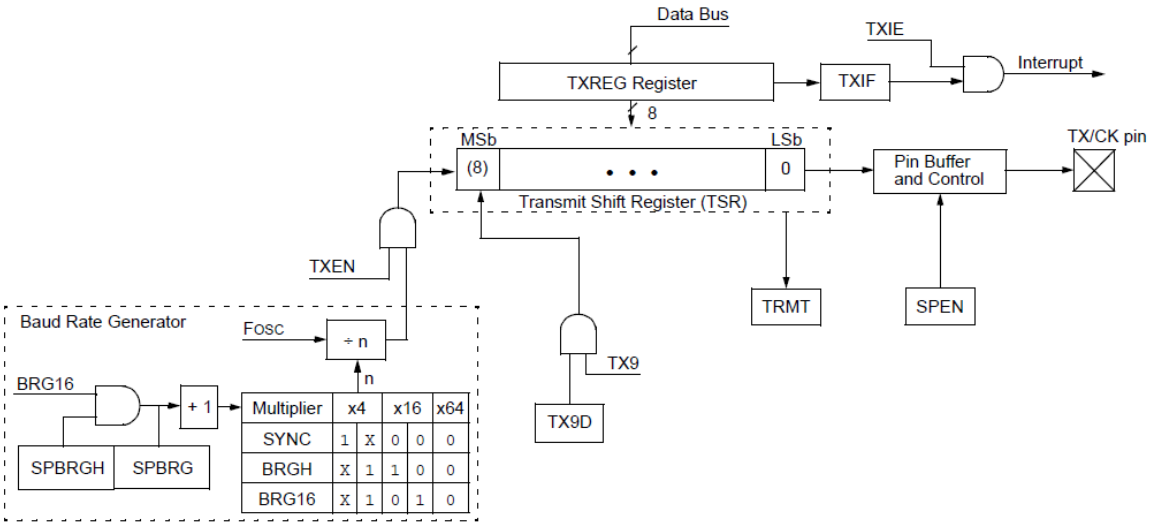


Figura 5.5. Esquema UART com a transmissor

5.1.7. Interrupcions

Una interrupció és una indicació de què hi ha hagut un esdeveniment en el hardware o en el software d'un ordinador, provocant, seguidament, una ruptura de la seqüència d'instruccions que realitza, passant a executar una rutina especial anomenada servei d'interrupció. Una vegada completada la rutina del servei d'interrupció, es retorna al punt del programa on s'ha produït la ruptura.

Una de les interrupcions utilitzades en aquest projecte és la generada pel *Timer0* del microcontrolador i s'han d'habilitar tant les globals com les dels perifèrics per poder utilitzar-les mitjançant els registres i bits del PIC *INTCONbits.GIE* i *INTCONbits.PEIE*, respectivament. Per tant, ha sigut útil consultar el diagrama de blocs del *Timer0*, *Figura 5.6*, per dissenyar el servei d'interrupcions del programa.

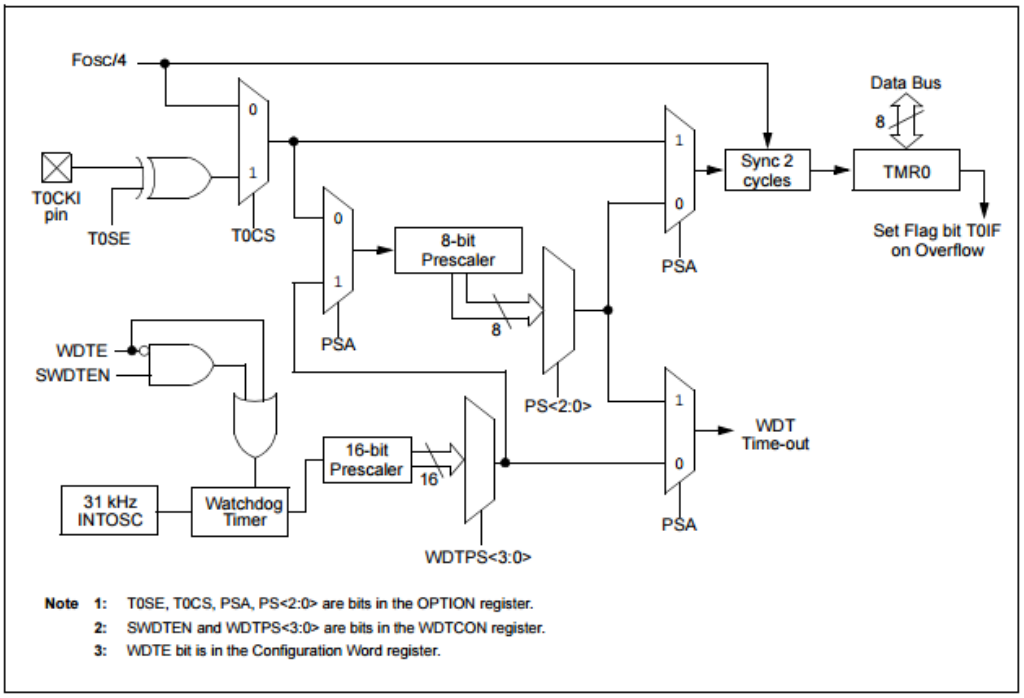


Figura 5.6. Diagrama de blocs del Timer0 del PIC16F690

A més, en el *Datasheet* del PIC16F690 es troba un diagrama de blocs de les interrupcions, *Figura 5.7*, molt útil a l'hora de programar el servei d'interrupcions, ja que en aquest diagrama s'aprecien les interrupcions que s'han d'habilitar. Per exemple, les mencionades *GIE* i *PEIE*, les interrupcions globals i les dels perifèrics, així com *T0IE*, per activar les interrupcions del *Timer0*, que, com ja s'ha dit, és el que les genera.

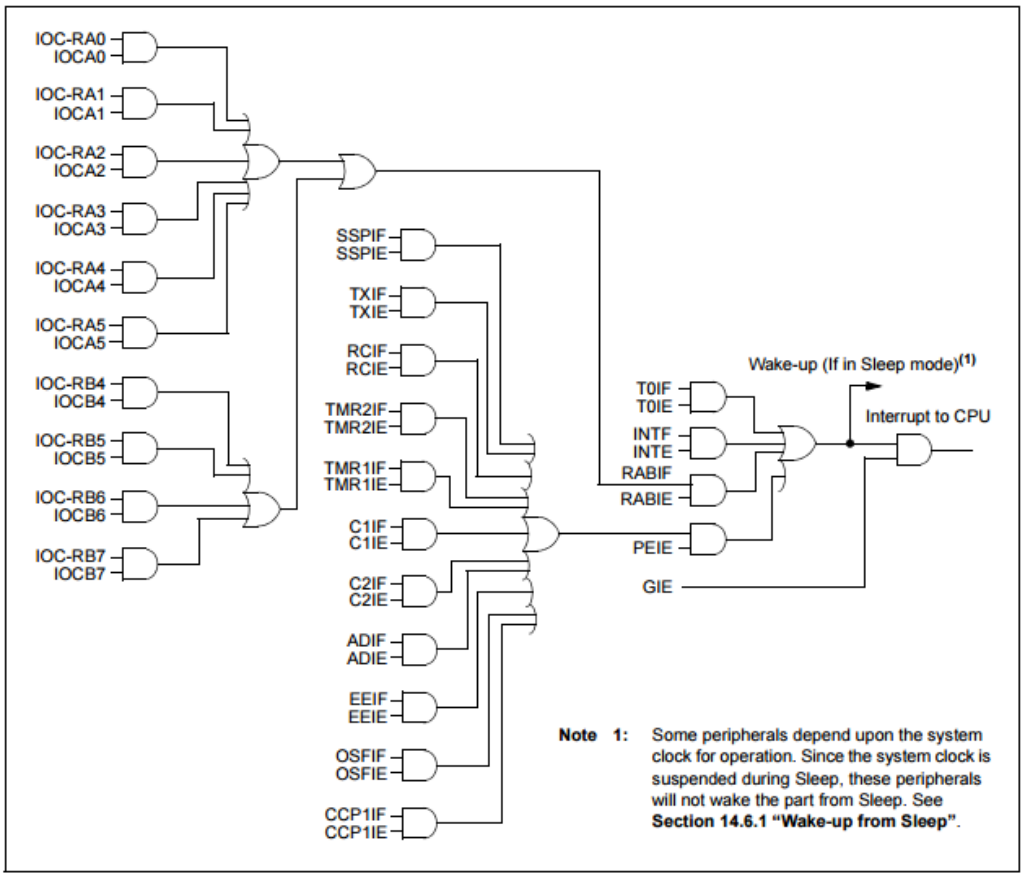


Figura 5.7. Diagrama de blocs de les interrupcions

5.1.8. MPLAB i PICkit2

Per tal de programar el microcontrolador han estat necessàries dues eines: el MPLAB (*software*) i el PICkit2 (*hardware*) **[2][3]**. El MPLAB és un editor IDE (de l'anglès *Integrated Development Enviroment*) gratuït que permet programar, entre d'altres, en llenguatge C. Aquest software és ofert per *Microchip Technology Inc*. El MPLAB permet editar un arxiu amb extensió .c i realitzar correccions. A més, conté un preprocessador que realitza manipulacions directives abans de compilar. Aquestes manipulacions han de començar amb "#" per tal de ser reconegudes. També consta d'un enllaçador que, tal com diu el seu nom, enllaça les llibreries referenciades amb el programa i crea un arxiu amb extensió .hex que es carrega a la memòria del microcontrolador. Un cop preprocessat, el compilador passa el codi C a codi llenguatge màquina, que és el codi que el microcontrolador pot executar (patrons de 1's i 0's).

Un cop fet i depurat el programa cal carregar-lo a la memòria ROM del microcontrolador. Per a fer-ho s'ha utilitzat el PICkit2 (*Figura 5.8*), que és un programador de baix cost que connecta l'ordinador amb el PIC i li transmet el programa.

Figura 5.8. PICkit2

A la *Figura 5.9* es mostren els pins del PICkit, que aniran connectats directament al microcontrolador.

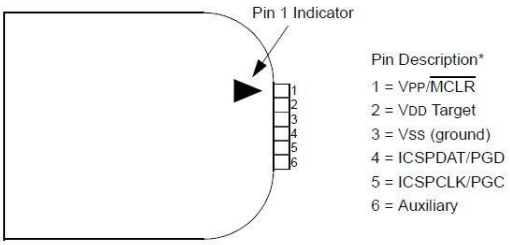


Figura 5.9. Pins del PICkit2

5.2. Sensor de color

El sensor de color utilitzat és un dispositiu convertidor de llum a freqüència. En concret, el model utilitzat en aquest treball és el TCS3210 **[4]**. La sortida que genera és una ona quadrada (cicle de treball 50%) amb una freqüència directament proporcional a la intensitat de llum, irradiància. El sensor permet seleccionar tres filtres diferents (*Red, Green, Blue*) sobre els fotodíodes i això permet obtenir informació sobre la intensitat de llum filtrada incident.

El sensor està muntat sobre una placa de circuit imprès que té incorporats quatre LEDs encarregats d'emetre la llum blanca que es projectarà sobre l'objecte de color que es vol caracteritzar. Filtra les dades RGB de la llum projectada que rep, convertint-les a una ona quadrada, que té la freqüència proporcional a la intensitat de la llum emesa. A més, té una matriu de 8x8 fotodíodes en el centre del component. D'aquests fotodíodes, 16 tenen filtres vermells, 16 els tenen verds, 16 tenen filtres blaus i, finalment, 16 sense filtre de color.

El component consta d'uns pins per tal de poder realitzar les connexions (*Figura 5.10*). Per començar, té dos pins, el S0 i el S1, amb els que es pot escalar la freqüència de sortida,

escollint entre el 2%, el 20% i el 100% de la freqüència màxima que podria generar. En aquest treball, d'acord als resultats que es volien obtenir, s'ha escollit la connexió del 2% de la freqüència. Seguidament, el sensor té dos pins, el S2 i el S3, pels quals es controla el filtre RGB. Per últim, té el pin d'alimentació, V_{DD}, el de terra, GND, un pin que permet habilitar i inhabilitar el sensor, $\overline{OE}$, i un pin de sortida, OUT.

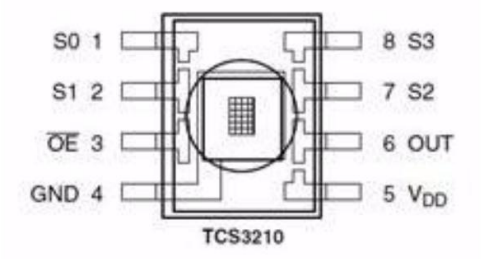


Figura 5.10. Diagrama de pins del sensor de color TCS3210

Per poder caracteritzar els colors i visualitzar les freqüències obtingudes, es connecta un oscil·loscopi al sistema (*Figura 5.11*) i se'n mesura la freqüència generada pel sensor davant d'objectes de diferents colors.

Figura 5.11. Connexions del sensor de color amb l'oscil·loscopi

Per tal de poder caracteritzar el màxim de colors i tons possibles, s'ha fet la caracterització dels colors de quinze cartolines diferents. A més, per aconseguir una caracterització més exacta, al sensor se li ha afegit una caixa negra, feta de cartolina d'aquest color, a fi de que la llum emesa es projecti únicament en el color desitjat.

El sensor de color disposa de quatre tipus de connexions segons el filtre RGB i d'acord amb els colors dels filtres dels diferents fotodíodes. Així doncs, té una pel vermell (*Red*),

una pel verd (*Green*), una pel blau (*Blue*) i una pel blanc (*Clear*), per tant, cada cartolina s'ha caracteritzat quatre vegades, una per cada tipus de connexió.

A la taula següent (*Taula 5.6*), es mostren els resultats de les diferents caracteritzacions, on s'indiquen el nom del color i les freqüències corresponents obtingudes experimentalment als tres tipus de connexions per aquella cartolina.

COLOR	FREQÜÈNCIA *RED* (Hz)	FREQÜÈNCIA *GREEN* (Hz)	FREQÜÈNCIA *BLUE* (Hz)	FREQÜÈNCIA *CLEAR* (Hz)
Blau clar	436,7	689,7	935,0	900,8
Blau	307,7	450,4	751,9	729,9
Blau marí	256,4	269,5	409,8	396,8
Verd clar	740,8	869,6	714,3	694,4
Verd	454,5	621,1	485,5	480,8
Verd fosc	288,2	687,6	396,8	393,7
Verd caqui	289,9	293,3	318,5	317,5
Vermell clar	757,5	310,6	381,1	381,7
Vermell	625,0	274,0	358,4	359,7
Granat	456,6	240,5	311,5	315,5
Groc clar	1124	925,9	625,0	606,1
Groc fosc	1136	769,3	512,8	505,0
Taronja	892,8	371,7	404,9	398,4
Lila	492,6	387,6	645,2	632,9
Blanc	1064	1053	1282	1316

Taula 5.6.Resultats de les caracteritzacions de colors

Per altra banda, a la *Taula 5.7* es mostren els valors en RGB, és a dir, valors entre el 0 i el 255. Per a poder fer el canvi de freqüència a RGB, s'ha fet una relació tenint en compte que el valor màxim de freqüència mesurada experimentalment, és a dir, la freqüència *clear* del blanc que és 1316Hz, era equivalent a 255. Per tant, els altres valors s'han multiplicat per 255 i s'han dividit per 1316Hz, arrodonint-los a la unitat més propera.

COLOR	*RED*	*GREEN*	*BLUE*	*CLEAR*
Blau clar	85	134	181	175
Blau	60	87	146	141
Blau marí	50	52	79	77
Verd clar	144	169	138	135
Verd	88	120	94	93
Verd fosc	56	133	77	76
Verd caqui	56	57	62	62
Vermell clar	147	60	74	74
Vermell	121	53	69	70
Granat	88	47	60	61
Groc clar	218	179	121	117
Groc fosc	220	149	99	98
Taronja	173	72	78	77
Lila	95	75	125	123
Blanc	206	204	248	255

Taula 5.7.Resultats de les caracteritzacions de colors en valors RGB

5.3. Mòdul *Bluetooth*

Un mòdul *Bluetooth* permet la recepció i transmissió d'informació sense la necessitat de cables ni connectors. Utilitza una radiofreqüència segura de 2,4 GHz i permet crear xarxes domèstiques sense fils útils per a compartir informació emmagatzemada en diferents dispositius.

En aquest projecte s'ha utilitzat el mòdul *Bluetooth* to Serial HC-05, del fabricant *ITead Studio* (*Figura 5.12)* **[5]**. La seva funció és transformar les dades en format decimal a codi binari de 8 bits per transmetre-les del mòbil al PIC, i de codi binari (1 byte) a decimal per transmetre-les del PIC al dispositiu mòbil.

Figura 5.12. Bluetooth to Serial HC-05

Aquest mòdul *Bluetooth*, tal com es pot observar a la *Figura 5.13*, consta de sis pins: STATE, RXD, TXD, GND, VCC i EN. STATE i EN no són utilitzats en aquest projecte.

Figura 5.13. Pins del mòdul Bluetooth HC-05

Els pins V_{CC} i GND (*Ground*) serveixen per subministrar energia elèctrica al mòdul *Bluetooth*. Aquest model en concret està dissenyat per rebre entre 3,3V i 5V. Els pins RXD i TXD són utilitzats per a la comunicació serial UART. RXD s'utilitza per a la recepció de dades i TXD per a la transmissió. Estan connectats als pins 10 i 12 del PIC, respectivament.

Com ja s'ha esmentat, els pins STATE i EN (*Enable*) no han estat utilitzats. STATE serveix per indicar quin és el mode de treball (mestre o esclau) i EN per accedir al mode de configuració del mòdul *Bluetooth*. EN permet configurar paràmetres com ara la velocitat de transmissió o el nom del mòdul *Bluetooth*.

En un model de comunicació mestre-esclau, hi ha un dispositiu (el mestre) que té control unidireccional sobre els altres (els esclaus). En un protocol de comunicació *Bluetooth* cal

que hi hagi un mestre i un o més esclaus. El model HC-05 pot actuar tant com a mestre com a esclau. En aquest projecte, el dispositiu mòbil actua com a mestre mitjançant l'*App*. Busca quins dispositius *Bluetooth* hi ha actius i s'emparella amb el desitjat. El mòdul *Bluetooth* actua, per tant, com a esclau.

Algunes de les característiques a tenir en compte del model *Bluetooth* to Serial HC-05 són les següents:

- Velocitats de transmissió (*Baud rate*) configurables: 9600, 19200, 38400 (per defecte), 57600, 115200, 230400, 460800
- Dades de 8 bits
- 1 bit de parada
- No té bit de paritat
- Disponibilitat de codi d'emparellament (0000 per defecte)
- Seqüències de LED indicant l'estat de connexió: Parpelleig del LED dues vegades per segon (desconnectat) i parpelleig del LED una vegada cada dos segons (emparellat)

5.4. Pantalla LCD

Per tal de poder visualitzar els resultats de les mostres preses pel sensor de color, s'ha implementat una pantalla LCD (de l'anglès *Liquid Crystal Display*). En aquest treball s'ha utilitzat el model Hitachi HD44780 controller de 16x2 **[6]**.

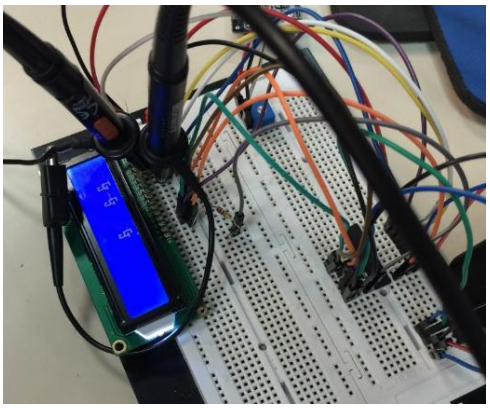


Figura 5.14. Connexions del microcontrolador i la pantalla LCD

La pantalla LCD va connectada a un microcontrolador, com es pot veure a la Figura 5.14, mitjançant el qual la LCD funciona d'acord la manera desitjada. El component consta de 16 pins (*Figura 5.15*), dels quals V_{ss}, connexió a terra, i V_{dd}, 5V, són per a alimentar-lo correctament. A més, també té el pin per alterar el contrast de la pantalla, V_{ee}, el qual ha estat ajustat amb una resistència variable, i tres pins per a les comandes de control. El

primer senyal de control és RS, *Register Select*, posat a 1 perquè es puguin transmetre caràcters, ja que si aquest registre es posa a 0, les dades rebudes pel dispositiu són enteses com ordres. Seguint amb els senyals de control, R/W, *Read/Write*, està posat a 1 per tal de què llegeixi la informació que rep i la mostri per la pantalla. L'últim d'aquests senyals és E, *Enable*, que serveix per activar o desactivar les funcions de la LCD. Seguidament, hi ha els pins D0 fins a D7, que corresponen al bus de dades, per transmetre i/o rebre informació. Finalment, hi ha els pins A i K, que són l'ànode i el càtode de la il·luminació de la pantalla, respectivament.

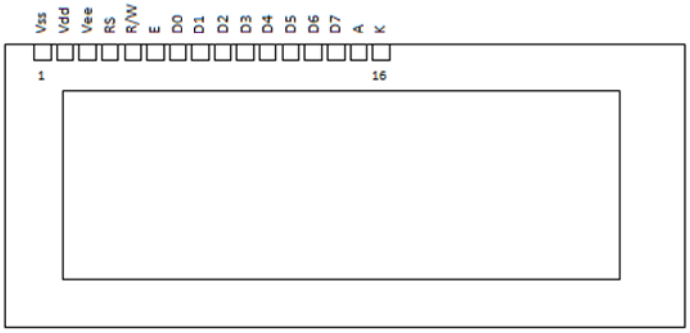


Figura 5.15. Diagrama de pins de la LCD HD44780

Com s'ha comentat, depenent si el senyal de control *Register Select* està posat a 0 o a 1 les dades que rep la LCD s'entenen com a ordres o es transmeten com a caràcters. En la taula següent, *Figura 5.16*, es mostren les ordres que pot rebre el dispositiu quan el RS està posat a 0 i què s'ha de posar en els pins D0 fins al D7 per poder transmetre aquestes ordres.

Command	Binary								Hex
	D7	D6	D5	D4	D3	D2	D1	D0	
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	x	02 or 03
Character Entry Mode	0	0	0	0	0	1	I/D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x	10 to 1F
Function Set	0	0	1	8/4	2/1	10/7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	40 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

I/D: 1=Increment*, 0=Decrement
S: 1=Display shift on, 0=Display shift off*
D: 1=Display On, 0=Display Off*
U: 1=Cursor underline on, 0=Underline off*
B: 1=Cursor blink on, 0=Cursor blink off*
D/C: 1=Display shift, 0=Cursor move
R/L: 1=Right shift, 0=Left shift
8/4: 1=8 bit interface*, 0=4 bit interface
2/1: 1=2 line mode, 0=1 line mode*
10/7: 1=5x10 dot format, 0=5x7 dot format*
x = Don't care
* = Initialisation settings

Figura 5.16. Conjunt d'ordres que pot rebre la LCD i com transmetre-les

El *Clear Display* consisteix en netejar la pantalla del dispositiu posant-la en blanc i col·locant el cursor en la primera posició. El *Display & Cursor Home* serveix per col·locar el cursor en la posició d'inici. El *Character Entry Mode* estableix la direcció del moviment del cursor. El *Display On/Off & Cursor* activa o desactiva tant el dispositiu (D) com el cursor (U), mostrant-lo subratllat, i estableix si aquest fa pampallugues o no (B). El *Display/Cursor Shift* mou el cursor i desplaça la visualització de la pantalla, segons s'indiqui, així com en el sentit que s'estableixi. El *Function Set* estableix la mida de la interfície, el nombre de files de la pantalla i el format del caràcter. El *Set CGRAM Address* s'utilitza per crear caràcters. Per acabar amb les ordres, el *Set Display Address* s'usa per determinar la posició del cursor.

En canvi, si el RS està a 1, la pantalla LCD transmet caràcters per a la seva visualització. A continuació, a la Figura 5.17, es mostren els caràcters estàndards que pot mostrar el dispositiu.

Figura 5.17. Taula de caràcters estàndars de la LCD

El funcionament de la pantalla ve determinat pel programa, realitzat amb C (*Annex 2*), segons el qual capta la informació obtinguda pel sensor de color i l'escriu per tal de poder visualitzar els resultats. Així, es mostra la quantitat de vermell (R), verd (G), blau (B) i blanc (*Clear*) que conté el color que s'està analitzant.

5.5. LED RGB

Un LED RGB (de les sigles en anglès *Red, Green, Blue*) és un díode emissor de llum basat en tres LEDs en un sol encapsulat dels colors primaris de la llum: un vermell, un verd i un blau. Mitjançant aquests LEDs, segons amb la intensitat amb la que s'encengui cada un, es pot mostrar qualsevol color. El model utilitzat en aquest projecte és el L154A4SURKQBDZGW **[7]**.

El RGB és un model de definició de colors amb el qual és possible representar un color mitjançant la mescla dels tres colors de llum primaris. Aquest model s'utilitza en treballs digitals i cada valor de color pren valors del 0 al 255. Així doncs, el vermell en RGB és el (255, 0, 0), el verd és el (0, 255, 0) i el blau el (0, 0, 255). El color blanc és la mescla dels tres colors. A la següent figura, *Figura 5.18,* es representa el model RGB.

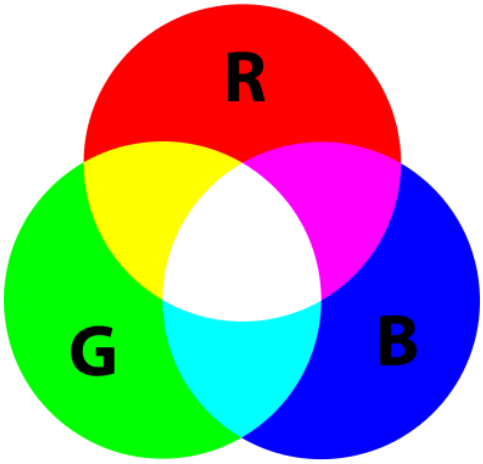


Figura 5.18. Representació del model RGB

En aquest treball, el LED RGB s'ha utilitzat per mostrar un color del qual s'ha escollit la composició de vermell, verd i blau a través de l'aplicació de mòbil *Android*.

6. Aplicació *Android*

Per a facilitar la interacció entre l'usuari i el sistema, s'ha dissenyat una aplicació per a un telèfon mòbil amb sistema operatiu *Android*. L'*App* té dues funcions: enviar la informació del color generat per l'usuari al microcontrolador per a què el color sigui emès pel LED i rebre informació del color captat pel sensor. La comunicació entre el dispositiu mòbil i el microcontrolador es fa via *Bluetooth*.

Per al disseny de l'aplicació s'ha utilitzat l'*App Inventor 2* **[8]**, que és un software gratuït dissenyat pel MIT (*Massachussetts Institute of Technology*). Aquesta plataforma, que ha estat desenvolupada per *Google Labs*, permet la creació i disseny d'aplicacions mòbils d'una manera senzilla i intuïtiva amb un llenguatge de programació a l'abast d'un públic poc familiaritzat en aquest àmbit.

6.1. MIT *App Inventor 2*

Primer de tot, mitjançant un compte de Google, cal crear un nou projecte. Un cop creat, s'obre la plataforma de treball, que consta de dos entorns: el dissenyador (*designer*) i els blocs (*blocks*). El dissenyador permet veure l'aparença de l'*App* i afegir els elements que es necessitin. L'apartat de blocs és on es programen les accions de cadascun dels elements.

El dissenyador, tal com es pot veure en la *Figura 6.1*, consta de diferents espais: paleta (*palette*), visor (*visor*), components (*components)*, mitjans (*media*) i propietats (*properties)*. A més, a dalt a la dreta hi ha dos botons que permeten canviar del dissenyador als blocs, i viceversa, i a l'esquerra, tres botons que permeten navegar entre les diferents pantalles de l'*App*, afegir-ne de noves o esborrar les no desitjades.

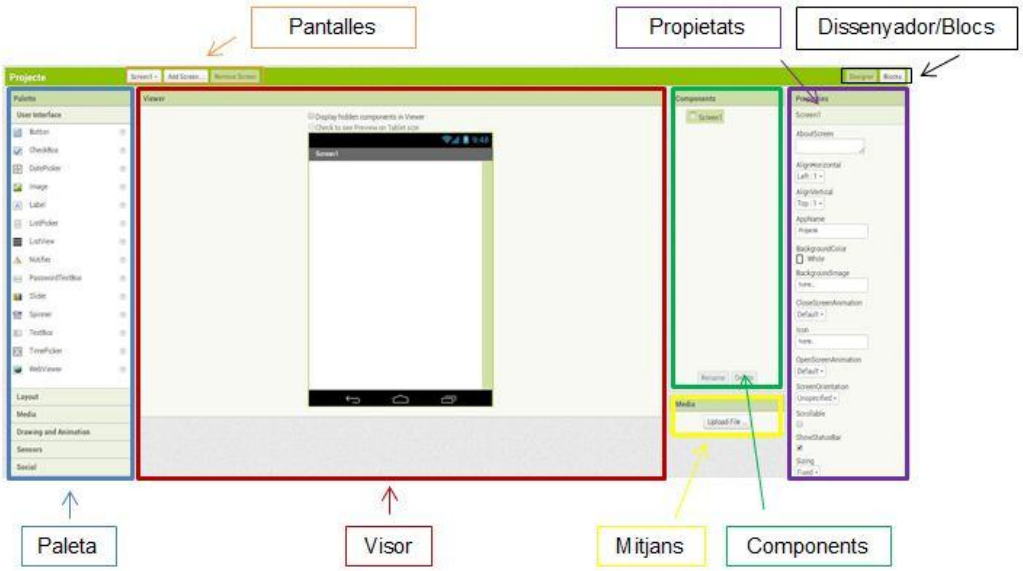


Figura 6.1. Aparença del dissenyador

A continuació, es fa una breu explicació d'aquests espais i què s'ha utilitzat de cadascun d'ells.

La **paleta**, que està subdividida en diferents subespais, permet seleccionar els elements necessaris per al disseny de l'*App*. Els subespais que s'han utilitzat són: interfície d'usuari (*user interface*), disposició (*layout*), sensors (*sensors*) i connectivitat (*connectivity*). A la *Figura 6.2*, s'assenyalen tots els elements que s'han fet servir de cadascun dels subespais de la paleta.

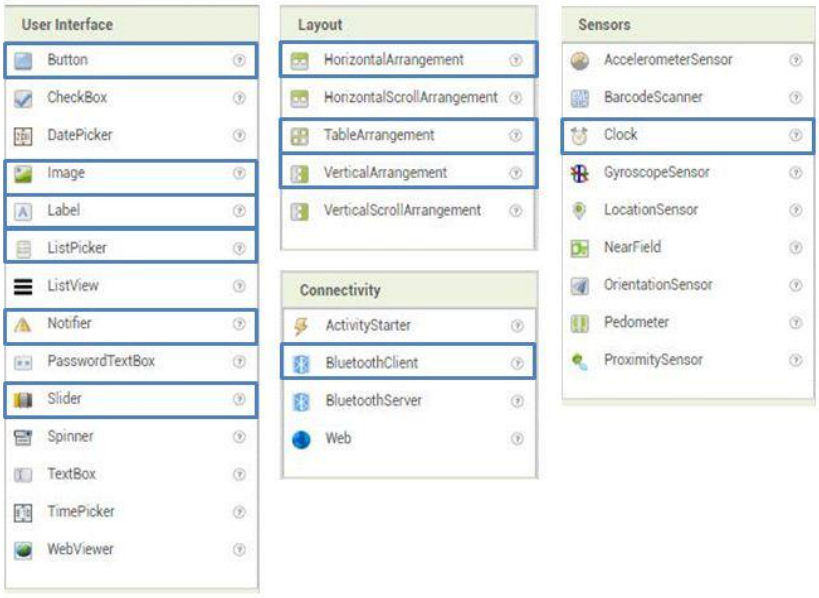


Figura 6.2. Elements utilitzats de la paleta classificats en subespais

Per a seleccionar els elements desitjats, simplement s'han d'arrossegar fins al visor i deixar-los anar sobre l'espai que simula la pantalla d'un telèfon mòbil. En la *Taula 6.1* s'esmenen els elements utilitzats de cada subespai i què permeten fer cadascun d'ells.

USER INTERFACE	
Element	**Funció**
Button	Al clicar-lo, permet la realització d'accions
Image	Permet mostrar fotografies
Label	Insereix un requadre on es pot afegir text
ListSelector	Botó que al ser pressionat mostra una llista de textos
Notifier	Permet mostrar quadres amb alertes o missatges temporals
Slider	Barra de progrés amb un marcador que pot ser desplaçat
LAYOUT	
Element	**Funció**
HorizontalArrangement	Permet organitzar elements horitzontalment
TableArrangement	Taula que permet l'organització d'elements
VerticalArrangement	Permet organitzar elements verticalment
CONNECTIVITY	
Element	**Funció**
BluetoothClient	Permet habilitar les funcions de Bluetooth
SENSORS	
Element	**Funció**
Clock	Temporitzador

Taula 6.1. Elements utilitzats de la paleta amb les seves respectives funcions

El **visor** permet veure quina serà l'aparença de l'aplicació i modificar-la. Quan un element és arrossegat sobre la pantalla del mòbil del visor s'afegeix automàticament a **components**, espai que permet veure quins elements hi ha seleccionats, modificar el seu nom o eliminar-los. Tots els elements apareixen en forma de llista a components però no tots ells són visibles en la pantalla del dispositiu mòbil. Per exemple, la connectivitat *Bluetooth* o el notificador són elements que tenen funcions dintre de l'*App* però que no són vistos per l'usuari en la pantalla.

En l'espai **propietats**, es poden configurar els elements seleccionats: color, mida, lletra, forma, i altres propietats específiques de cada element. Per penjar un arxiu de so, imatge o vídeo s'utilitza l'espai **mitjans**.

Tal com s'ha esmentat prèviament, fent ús dels botons de dalt a la dreta, es pot canviar de l'entorn del dissenyador al dels blocs. Aquest, tal com es pot observar a la *Figura 6.3*, té una aparença semblant a la del dissenyador.

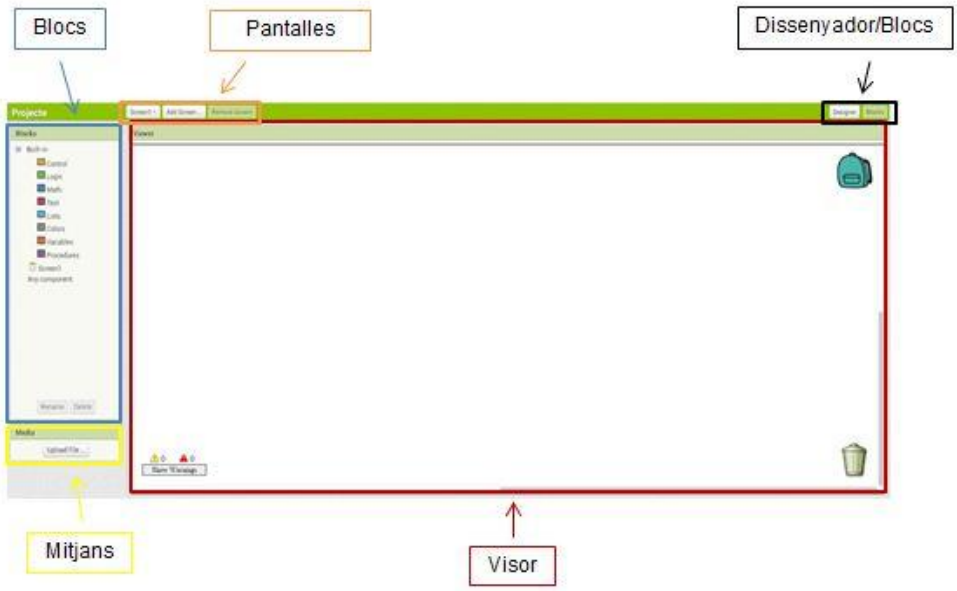


Figura 6.3. Aparença entorn blocs

Aquest, igual que en el cas del dissenyador, té diferents espais: blocs (*blocks)*, mitjans (*media*) i visor (*visor*). A més, continua havent-hi els botons per gestionar les diferents pantalles de l'*App* i els botons per canviar a l'entorn del dissenyador.

L'espai **blocs** consta de diferents subespais classificats per tipus de blocs. Això fa que quan es necessita una certa acció, sigui força intuïtiu trobar-la. A més, també apareixen les pantalles que s'han creat amb els elements que s'han seleccionat a l'apartat del dissenyador, els quals tenen, també, les seves pròpies comandes. Quan es fa clic sobre un dels tipus de blocs, es desplega una pestanya amb les possibles accions que pot realitzar. S'arrosseguen fins al visor i s'encaixen les unes amb les altres a mode de trencaclosques. A la *Figura 6.4* es pot veure l'espai blocs amb tots els seus subespais (dins el menú *Built-in*) i, a mode d'exemple, la pestanya de control desplegada amb les comandes que hi ha disponibles.

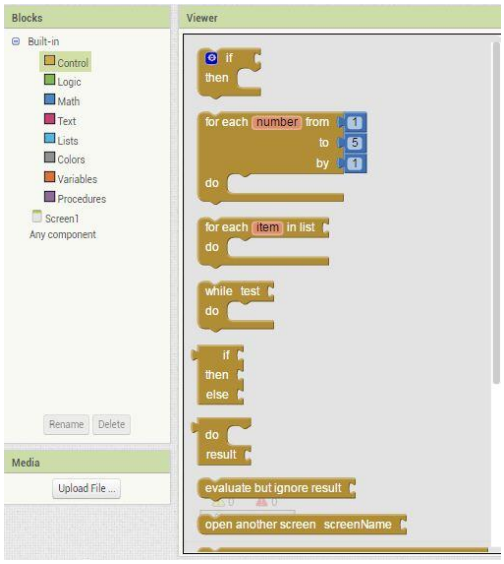


Figura 6.4. Espai blocs amb la pestanya "Control" desplegada

L'espai **visor** s'utilitza per a treballar amb els blocs i crear el programa. L'espai de **mitjans**, igual que en l'entorn del dissenyador, s'utilitza per penjar arxius de so, imatge o vídeo.

A continuació, s'explica detalladament els blocs que s'han utilitzat i quines són les seves funcions.

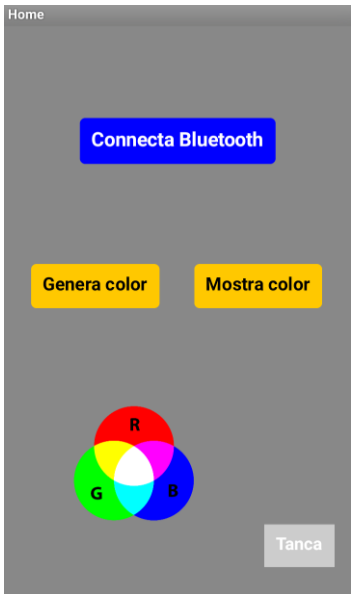
6.2. Programa implementat en el projecte

En aquest projecte, la creació d'una *App* pretén facilitar l'enviament i rebuda de dades entre el microcontrolador i el dispositiu mòbil. Així doncs, té dues funcions principals: que l'usuari, via *Bluetooth*, pugui enviar dades sobre un color que hagi generat des del dispositiu mòbil i reproduir-lo en un LED RGB, i que l'usuari pugui veure reproduït per la pantalla el color captat pel sensor i en pugui saber les tres components RGB.

L'*App* consta de tres pantalles anomenades Screen1, GENERA_COLOR i MOSTRA_COLOR. Per tal de fer més entenedor el programa, s'han escrit els noms de les pantalles en majúscules i el nom dels botons en minúscules.

L'Screen1 (*Figura 6.5*) és la pantalla principal, la que l'usuari es troba a l'iniciar l'aplicació. Aquesta, té un selector de llista (Connecta Bluetooth) i tres botons:

- Connecta *Bluetooth* (connexio_Bluetooth): Quan es clica s'obre una llista dels dispositius *Bluetooth* que hi ha actius a la zona i l'usuari pot triar amb quin es vol emparellar.



- "Genera color" (genera_color): Quan es clica s'obre una altra pantalla on l'usuari pot generar el color que desitgi i enviar-lo al microcontrolador.

- "Mostra color" (mostra_color): Quan es clica s'obre una pantalla amb el fons del color que el sensor està captant i indica les seves components RGB.

- "Tanca" (Sortir): Quan es clica se surt de l'aplicació.

Figura 6.5. Aparença Screen1

Els blocs utilitzats per aquesta pantalla són els que s'expliquen a continuació. En la *Figura 6.6* i *Figura 6.7*, es poden observar les instruccions necessàries per establir la connexió *Bluetooth*.

Quan l'usuari encèn el *Bluetooth* del seu telèfon i inicia l'aplicació, abans de clicar el selector de llista connexió_Bluetooth, es defineixen en la llista tots els dispositius que hi ha a l'abast (*Figura 6.6*).

Figura 6.6. Configuració llista dispositius Bluetooth

El selector de llista té l'aparença d'un botó. Quan es clica sobre "Connecta *Bluetooth*", s'obre una llista dels dispositius *Bluetooth* actius a l'abast del telèfon mòbil de l'usuari i s'ha de triar amb quin d'ells es vol emparellar. Un cop seleccionat, es realitza la connexió. Quan aquesta s'ha efectuat, es mostra en pantalla el missatge "Connectat!" i apareix el botó "Acceptar" per fer desaparèixer el missatge (*Figura 6.7*).

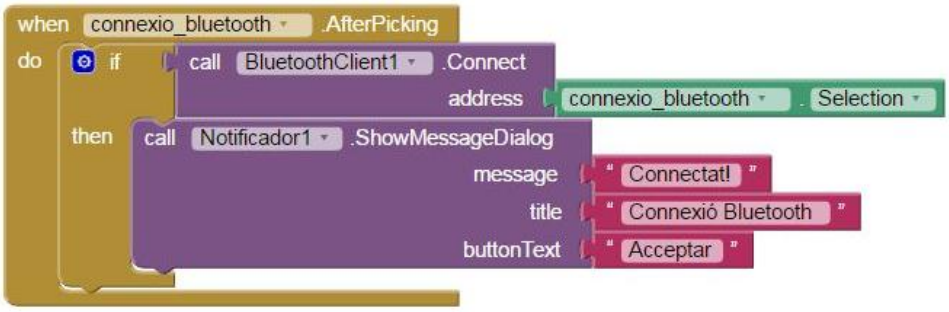


Figura 6.7. Connexió Bluetooth

Tal com es pot veure en la *Figura 6.8*, quan es fa clic sobre el botó "Genera color", es desconnecta el *Bluetooth* de la pantalla actual (Screen1) i s'obre la pantalla GENERA_COLOR agafant com a valor inicial el dispositiu *Bluetooth* que l'usuari havia prèviament seleccionat. Quan es canvia de pantalla, és imprescindible desconnectar el *Bluetooth* de la pantalla que es tanca i obrir-lo de nou a la nova pantalla. Amb aquesta ordre el que es fa és desconnectar-lo de la pantalla actual i preparar-ho perquè quan s'inicialitzi la nova pantalla pugui ser cridat fàcilment. Quan es fa clic sobre "Mostra color", s'efectua la mateixa operació però obrint la pantalla MOSTRA_COLOR. Per últim, quan es clica "Tanca", se surt de l'aplicació.

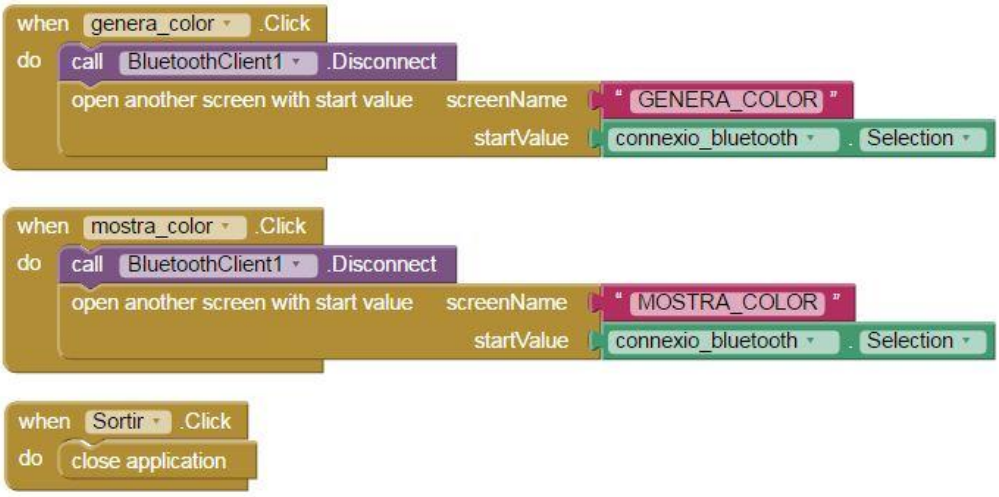


Figura 6.8. Accions dels botons de la pantalla Screen1

Un cop el mòbil està connectat amb el circuit via *Bluetooth*, ja es pot començar a enviar i rebre dades.

6.2.1. Generació del color sobre un LED

Per tal de realitzar la funció de generar color, l'usuari ha de clicar el botó "Genera color" (genera_color) i s'inicialitza la pantalla GENERA_COLOR, que té l'aparença que es mostra en la *Figura 6.9*.

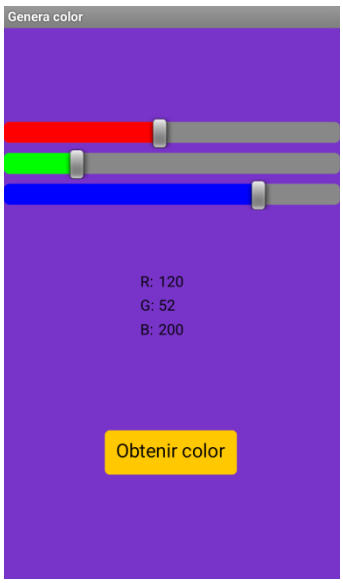


Figura 6.9. Aparença pantalla GENERA_COLOR

Consta dels següents elements:

- Tres lliscadors: R, G i B. Corresponen al lliscador vermell, verd i blau, respectivament. L'usuari pot fer lliscar cap a la dreta o cap a l'esquerra el marcador de cada un d'ells. A l'esquerra de tot val 0 i a la dreta de tot val 255.

- Sis etiquetes: red, resultat_red, green, resultat_green, blue i resultat_blue. El valor de les etiquetes red, green i blue és el mateix tota l'estona i és R:, G: i B:. El valor de resultat_red, resultat_green i resultat_blue serà un número entre el 0 i el 255 i anirà variant segons la posició del marcador de cada lliscador.

- Un botó: "Obtenir Color" (obtenir_color). Quan l'usuari el prem, s'envia al microcontrolador les dades RGB del color actual segons la posició dels lliscadors.

El color del fons de pantalla varia segons la posició dels lliscadors per tal que l'usuari pugui veure quin és el color que està generant. Al principi, el fons de pantalla és gris perquè tots els lliscadors es troben en una posició intermèdia (125). En la *Figura 6.9*, el fons de pantalla és el corresponent al color R: 120, G: 52 i B: 200.

En la *Figura 6.10* es mostren les accions que s'executen quan l'usuari clica el botó "Genera color" de la pantalla Screen1. Quan ho fa, s'inicialitza la pantalla GENERA_COLOR i, abans que l'usuari faci res en la nova pantalla, es realitzen una sèrie d'accions. El primer que es fa és efectuar la connexió *Bluetooth* amb el valor que prèviament s'havia establert com a valor inicial. A continuació, s'estableix que les etiquetes resultat_red, resultat_green i resultat_blue prenguin els valors actuals dels lliscadors R, G i B, respectivament. Per últim, s'estableix com a fons de pantalla un color generat per una llista de tres elements. Aquests tres elements han de ser números entre el 0 i el 255. En aquest cas, s'ha posat com a elements la posició dels lliscadors R,G i B.

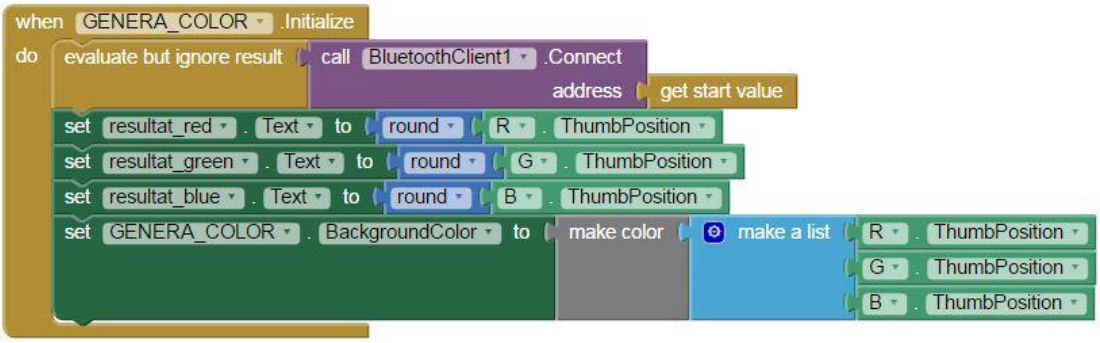


Figura 6.10. Instruccions per a la inicialització de la pantalla GENERA_COLOR

Com es pot veure en la *Figura 6.11*, quan l'usuari mou el marcador del primer lliscador (lliscador vermell, R) cap a la dreta o cap a l'esquerra, s'actualitza el color de fons de pantalla i també el valor de l'etiqueta resultat_red. D'aquesta manera, el fons de pantalla i el valor de les etiquetes correspondrà amb el nou color RGB.

Figura 6.11. Canvi de posició del lliscador R

Anàlogament, es fa la mateixa crida per als lliscadors G i B (*Figura 6.12*).

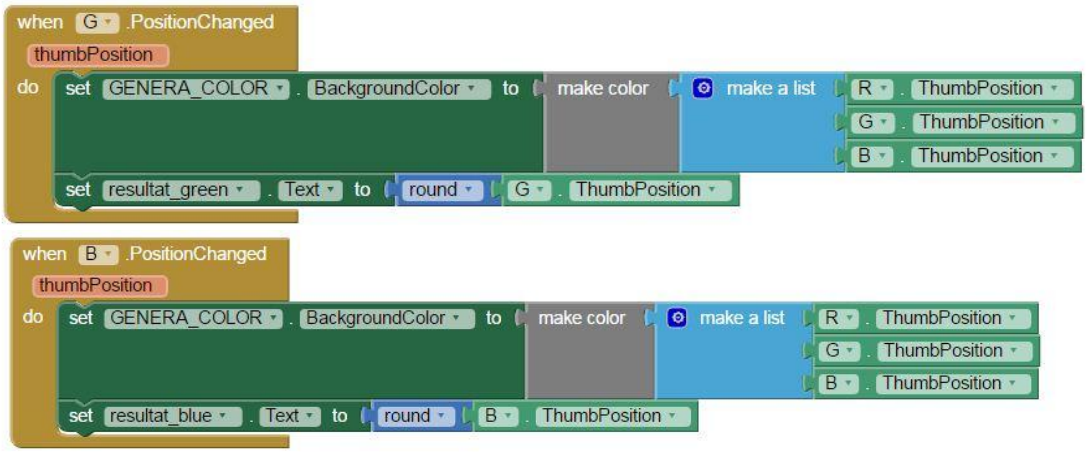


Figura 6.12. Canvi de posició dels lliscador G i B

Un cop obtingut el color desitjat, mitjançant el botó “Obtenir color”, l’usuari pot enviar les dades al microcontrolador i aquest al LED RGB per tal de reproduir el color generat.

Quan es clica “Obtenir color” (*Figura 6.13),* s’envia via *Bluetooth* una llista amb quatre elements. El primer, un 1, serveix per avisar que, a continuació, vindran tres dades que hauran de ser llegides per l’UART. Pel que fa als altres tres elements, cadascun és la posició d’un lliscador (un número entre el 0 i el 255). Aquesta llista és processada pel microcontrolador i s’encén el LED RGB mostrant el color desitjat.

Figura 6.13. Accions executades al prémer obtenir_color

6.2.2. Visualització dels components del color

Per tal de realitzar la funció de mostrar el color captat pel sensor, l'usuari ha de clicar el botó "Mostra color" (mostra_pantalla) i s'inicialitzarà la pantalla MOSTRA_COLOR, que té l'aparença que es mostra en la *Figura 6.9*.

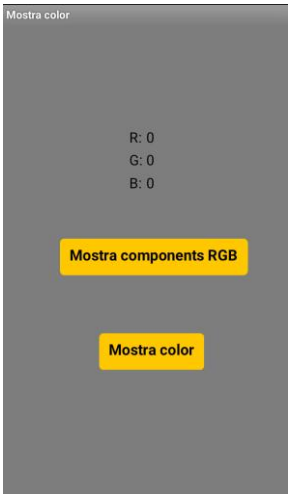


Figura 6.14. Aparença de la pantalla MOSTRA_COLOR

Consta dels següents elements:.

- Sis etiquetes: red, resultat_red, green, resultat_green, blue i resultat_blue. El valor de les etiquetes red, green i blue és el mateix tota l'estona i és R:, G: i B:. El valor de resultat_red, resultat_green i resultat_blue serà un número entre el 0 i el 255 i anirà variant segons el color captat pel sensor. Inicialment les tres variables valen 0 (tal com es mostra en la *Figura 6.14*)
- Dos botons: "Mostra components RGB" (mostra_RGB) i "Mostra color" (mostra_color). El primer, quan és premut, mostra les components RGB del color captat pel sensor mitjançant les etiquetes resultat_red, resultat_green i resultat_blue. El segon, "Mostra color", fa que el color del fons de pantalla canviï de color i passi a ser el color amb les components RGB captades pel sensor.

Abans que l'usuari cliqui sobre algun botó d'aquesta pantalla, s'inicialitzen les variables resultat_red, resultat_green i resultat_blue a 0. A més, s'efectua la connexió *Bluetooth* amb el valor que prèviament s'havia posat com a valor inicial al fer clic sobre el botó "Mostra color" i s'envia un 0 per tal d'avisar que el dispositiu està preparat per rebre dades. Per últim, es genera el color gris i es posa com a fons de pantalla.

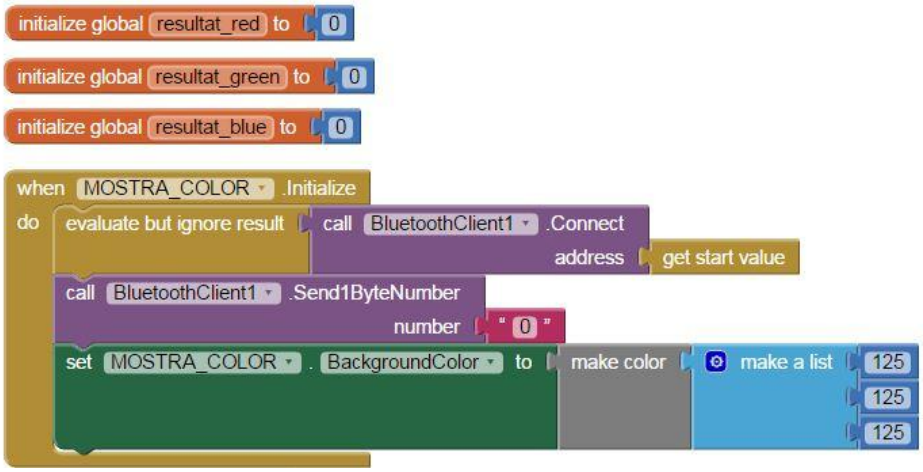


Figura 6.15. Instruccions per a la inicialització de la pantalla GENERA_COLOR

Un cop acabades les accions de la inicialització, si l'usuari clica el botó "Mostra components RGB" (mostra_RGB) es comprova, via *Bluetooth,* si es tenen bytes per rebre. Si el sensor està captant colors i el microcontrolador ha rebut correctament el 0 enviat a la inicialització de la pantalla, hi haurà bytes pendents de ser enviats. En concret, hi haurà 3 bytes, un per cada component RGB. Aquests bytes, al ser rebuts, s'emmagatzemen en les variables resultat_red, resultat_green i resultat_blue, respectivament. Les etiquetes passen a tenir el valor de cadascuna d'aquestes variables, permetent d'aquesta manera que l'usuari pugui llegir les components RGB per la pantalla del telèfon mòbil (*Figura 6.16)*.

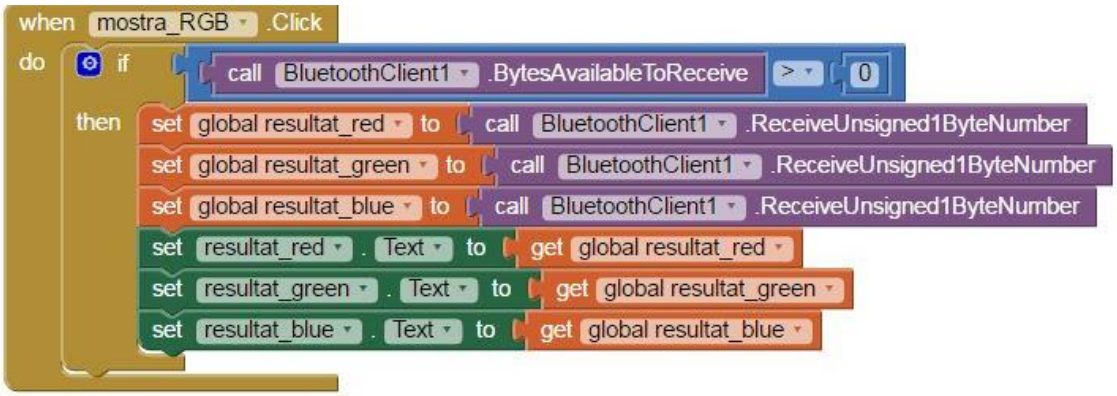


Figura 6.16. Accions que es realitzen al clicar mostra_RGB

Si es desitja veure el color captat pel sensor per la pantalla del telèfon mòbil, s'ha de clicar a "Mostra color" (mostra_pantalla). Al fer-ho, tal com es pot veure en la *Figura 6.17*, el color de fons de pantalla passa a ser el format per les tres variables resultat_red, resultat_green i resultat_blue, que valen les components RGB captades pel sensor.

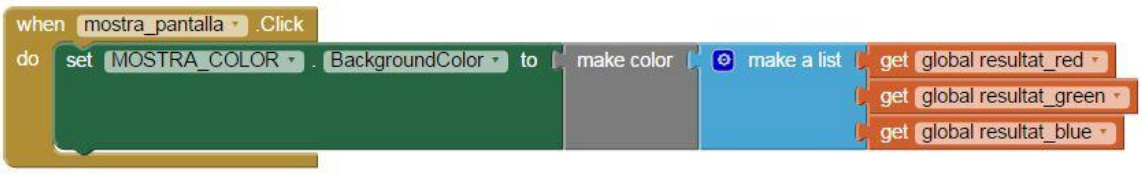


Figura 6.17. Acció realitzada al clicar el bóto Mostra color

És important que, abans de voler visualitzar el color captat com a fons de pantalla, es cliqui a "Mostra components RGB" per tal que les variables resultat_red, resultat_green i resultat_blue s'actualitzin amb els valors que s'estan captant en aquell moment pel sensor.

Cada cop que es vulguin veure les components RGB d'un color diferent caldrà tornar a l'Screen1 i tornar a clicar "Mostra color" per tal d'enviar un 0 i avisar al microcontrolador que es volen rebre dades.

7. Implementació del hardware

Per tal de complir amb les especificacions del sistema exposades al capítol 4, s'han muntat dos circuits diferents: un per a la connexió amb el dispositiu mòbil i un altre per a la connexió amb la LCD.

Abans de construir-los, però, cal estudiar com funciona cadascun dels components utilitzats i familiaritzar-se amb les eines de treball com són el software MPLAB o l'oscil·loscopi. En aquest apartat, s'expliquen primer algunes proves que s'han fet prèvies a la construcció i, a continuació, quines són les connexions dels components dels circuits finals.

7.1. Proves prèvies

Un cop estudiats els components que es necessitaven per al projecte, el primer que es va fer és caracteritzar el sensor. És necessari conèixer com mesura les freqüències per tal d'entendre i analitzar millor els resultats. L'estudi i caracterització d'aquest és el que s'explica al capítol 5.2. Un cop caracteritzat el sensor, es va voler assegurar el correcte ús de la pantalla LCD. Per tal de fer-ho, es van realitzar un seguit de proves. En aquestes proves es va veure que, a vegades, la pantalla no implementa el programa adequadament, de manera que la segona línia de la pantalla no s'aconsegueix llegir. Tot i així, aquest problema es soluciona apagant el microcontrolador i tornant-lo a encendre, posteriorment.

Al llarg de les proves de la pantalla, es va decidir fer un programa que dibuixés un caràcter determinat i el mostrés a la pantalla en diverses posicions d'aquesta, com es pot veure en la figura següent (*Figura 7.1*).

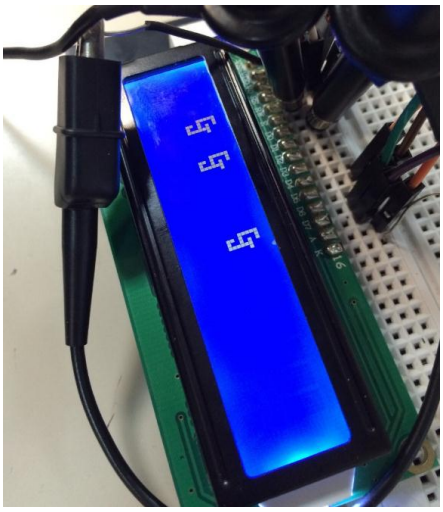


Figura 7.1.Caràcter dibuixat a la pantalla LCD, en diferents posicions d'aquesta

Posteriorment, es van realitzar unes proves per controlar la pantalla LCD amb l'aplicació per a mòbils. Es va crear una aplicació senzilla que enviava informació cap al microcontrolador. En la *Figura 7.2* es pot veure com, segons la informació que s'envia des de l'aplicació, o 10 o 40, en la pantalla s'observa 10 o 40, respectivament, junt amb els símbols de les proves anteriors. D'aquesta manera, a part d'estudiar el comportament de la LCD i de l'aplicació Android, també es va poder analitzar el comportament del mòdul *Bluetooth* i de l'UART.

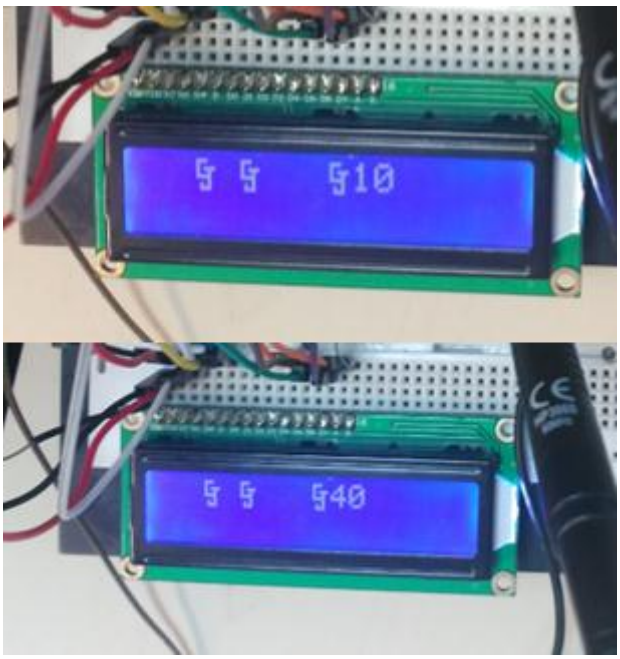


Figura 7.2. Proves de controlar la pantalla LCD amb l'aplicació per a mòbils

Un cop conegut el funcionament d'aquests components, es van començar a escriure els programes i a construir els circuits definitius. Les funcions dels programes es van anar fent per parts i es van anar provant cadascuna d'elles per entendre bé que succeïa en cada moment. Va ser molt útil la utilització de LEDs per entendre bé i comprovar què retornaven les funcions i també l'ús de l'oscil·loscopi.

7.1.1. Programació funcionalitat PWM

El microcontrolador té un mòdul PWM, de l'anglès *Pulse-Width Modulation*, que és un perifèric que modifica el cicle de treball d'una ona quadrada per tal de controlar la quantitat d'energia que envia. La modulació esmentada es pot veure en la següent figura, *Figura 7.3*.

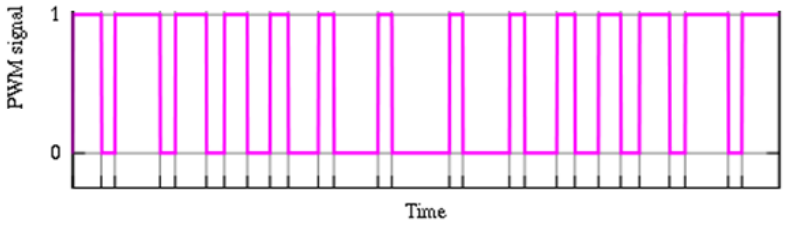


Figura 7.3. Gràfic d'un senyal PWM

Tot i que el microcontrolador, en aquest mòdul, té una funció PWM, en aquest projecte són necessàries tres funcions d'aquest tipus per a poder encendre el LED RGB de la manera desitjada en cada moment, per a poder controlar la quantitat de llum vermella, verda i blava que s'emet. Per tant, per a poder realitzar aquest control, a l'hora de la programació, s'ha creat una funció pròpia amb la funcionalitat del PWM, la qual funciona mitjançant interrupcions. Aquesta funció està explicada detalladament al capítol 8. El seu objectiu principal és que, donades les amplituds dels polsos *red*, *green* i *blue,* modula quanta estona haurà d'estar encès el vermell, verd i blau, respectivament, per tal de crear el color RGB desitjat. Al circuit final, les amplituds dels polsos es calculen a partir de la informació captada pel sensor. Tot i així, per provar si la funció creada funcionava bé, es van fer proves donant diferents valors als tres polsos i comprovant que el color que generava el LED RGB era el desitjat.

Les primeres proves que es van fer van ser reproduir els colors vermell, verd i blau per separat. Per a què el LED RGB doni llum exclusivament d'un d'aquests tres colors, l'amplitud del pols d'aquest ha de ser 100 (tota l'estona encès) i l'amplitud dels altres dos colors ha de ser 0.

Per a una amplitud de 100 per al vermell i 0 per al verd i el blau, el color resultant és el que es mostra a la *Figura 7.4.*

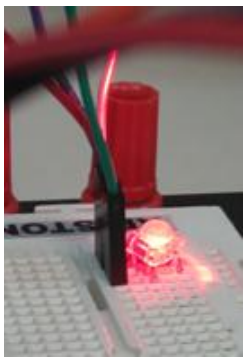


Figura 7.4. LED RGB amb amplitud red de 100 i amplitud green i blue de 0

Per a una amplitud de 100 per al verd i 0 per al vermell i el blau, el color resultant és el que es mostra a la *Figura 7.5.*

Figura 7.5. LED RGB amb amplitud green de 100 i amplitud green i blue de 0

Per a una amplitud de 100 per al blau i 0 per al vermell i el verd, el color resultant és el que es mostra a la *Figura 7.6.*

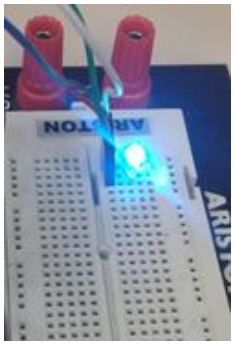


Figura 7.6. LED RGB amb amplitud blue de 100 i amplitud red i green de 0

Si en comptes de posar una amplitud a 100 i les altres a 0 es posa, per exemple, una a 50 i les altres dues a 0, el color obtingut és el mateix però amb una intensitat menor.

Si, mitjançant l'oscil·loscopi, s'analitza la senyal del color vermell quan la seva amplitud és de 100 i la del verd i el blau a 0, es veu que la seva ona sempre es manté a nivell alt (Figura 7.7).

Figura 7.7. Ona de sortida del vermell quan la seva amplitud és 100 i la del verd i blau és 0

A continuació, es continuen realitzant més proves. Si es posen totes les amplituds a 100, el color resultant és, tal com es pot veure en la *Figura 7.8*, el blanc.

Figura 7.8. Color resultant amb les amplituds dels tres colors a 100

Tot seguit, es va seguir provant fent barreges de colors. Amb una amplitud del vermell de 0 i una amplitud 100 per al blau i el verd, el color resultant és el de la *Figura 7.9*. Es tracta d'una barreja entre el verd i el blau.

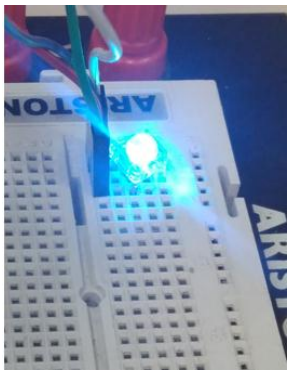


Figura 7.9. Color resultant d'una amplitud red de 0 i una amplitud green i blue de 100

Per últim, es prova una amplitud del vermell de 75, una amplitud del verd de 0 i una amplitud del blau de 50. Amb l'oscil·loscopi, s'observa el senyal de sortida del vermell i del blau. Tal com es pot veure a la *Figura 7.10*, l'amplitud de l'ona del vermell (senyal de dalt) està en nivell alt (LED encès), durant tres quartes parts del període de l'ona i l'amplitud de l'ona del blau (senyal de sota) està la meitat del període en nivell alt i l'altre meitat a 0.

Figura 7.10. Senyals de l'ona del vermell i del blau quan les seves amplituds són 75 i 50, respectivament

Amb aquestes proves, es va poder comprovar que el funcionament de la funció PWM dissenyada per tal d'encendre el LED RGB era l'esperat.

7.2. Connexions dels components

Per al desenvolupament d'aquest projecte s'han dissenyat dos circuits diferents: un per a la connexió amb el dispositiu mòbil i un altre per a la connexió amb la LCD. En aquest apartat, primer s'expliquen les interconnexions entre els diferents components utilitzats i el microcontrolador i, a continuació, es mostren les connexions completes d'ambdós circuits.

7.2.1. PICkit2

El programador PICkit2 consta de sis pins. En la *Taula 7.1* es mostra cada un dels pins del PICkit, quina és la seva funció i amb quin pin del microcontrolador estan connectats. A la *Figura 7.11* es pot veure l'esquema de connexions entre ambdós components.

Pin programador	Funció	Pin microcontrolador
1	Vpp/MCLR	4
2	Vdd	1
3	GND	20
4	ICSPDAT/PGD	19
5	ICSPCLK/PGC	18
6	Auxiliary	-

Taula 7.1.Relació pins PICkit2 - PIC16F690

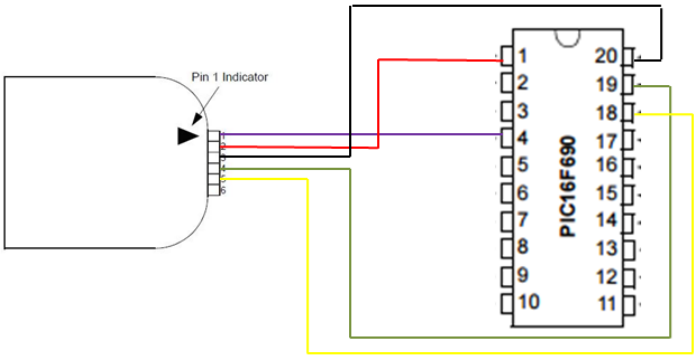


Figura 7.11. Connexions entre el programador i el microcontrolador

7.2.2. Sensor de color

El sensor de color utilitzat, el TCS3210, té vuit pins. En la *Taula 7.2* es mostra la funció de cadascun d'aquests pins i amb quin pin es connecten al microcontrolador. L'esquema de connexions és el de la *Figura 7.12*. Els pins S0, S1, S2, S3 i OUT es connecten als pins 19,18,17, 3 i 2, respectivament. La selecció d'aquests pins es definirà posteriorment al programa implantat al programador.

Pin sensor color	Funció	Pin microcontrolador
1	S0	19
2	S1	18
3	$\overline{OE}$	20
4	GND	20
5	Vdd	1
6	OUT	2
7	S2	17
8	S3	3

Taula 7.2. Relació pins sensor color - microcontrolador

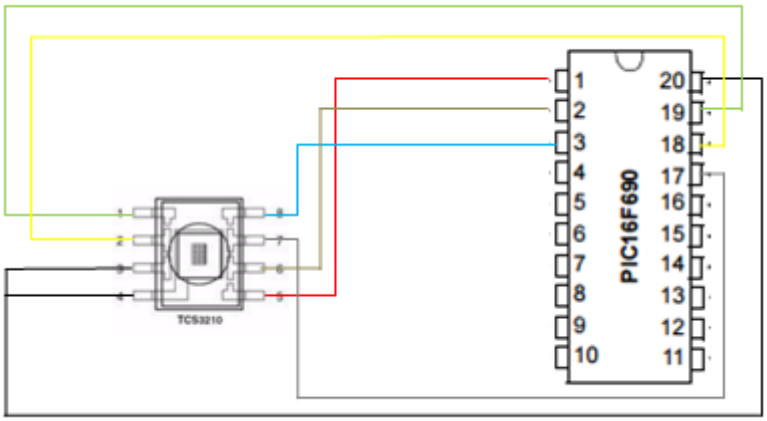


Figura 7.12. Connexions entre el sensor de color i el microcontrolador

7.2.3. Mòdul *Bluetooth*

El mòdul *Bluetooth* té sis pins. Per tal que el mòdul UART funcioni cal connectar els pins RX i TX del mòdul *Bluetooth* amb els de l'UART de forma creuada. És a dir, la recepció (RX) d'un amb la transmissió (TX) de l'altre i viceversa. Tal com s'especifica en el capítol 5.3, els pins STATE i ENABLE no s'utilitzen en aquest projecte. En la *Taula 7.3* es mostra cada un dels pins del mòdul *Bluetooth*, quina és la seva funció i amb quin pin del

microcontrolador estan connectats. A la *Figura 7.13* es pot veure l'esquema de connexions entre ambdós components.

Pin mòdul *Bluetooth*	Funció	Pin microcontrolador
1	STATE	-
2	RXD	12
3	TXD	10
4	GND	20
5	Vcc	1
6	EN	-

Taula 7.3. Relació pins mòdul Bluetooth – microcontrolador

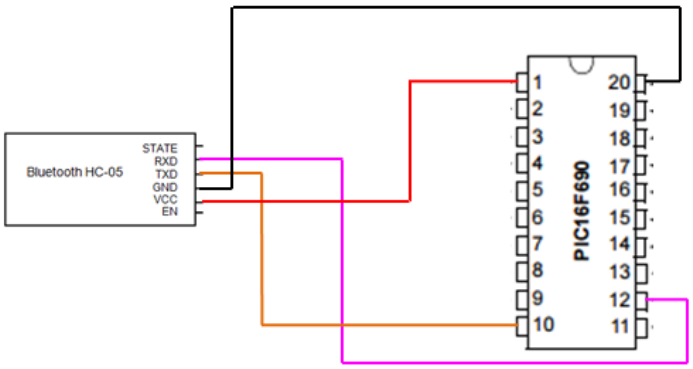


Figura 7.13. Connexions entre el mòdul Bluetooth i el microcontrolador

7.2.4. LED RGB

En aquest projecte s'han utilitzat dos LEDs RGB, un per a cada circuit. En els dos casos, el model utilitzat ha sigut el L-154A4ASURKQBDZGW que té quatre pins. D'aquests pins, un és la seva connexió a terra (GND) i els altres són la component vermella (R), la verda (G) i la blava. A la *Taula 7.4* es mostra la funció que té cada pin, així com els pins del microcontrolador en els que es connecten per cada circuit. Per altra banda, a la *Figura*

7.14, es poden veure esquematitzades aquestes connexions.

Pin LED RGB	Funció	Pin microcontrolador Circuit 1	Pin microcontrolador Circuit 2
1	GND	20	20
2	R	8	13
3	G	15	12
4	B	9	11

Taula 7.4. Relació pins LED RGB – microcontrolador per als dos circuits

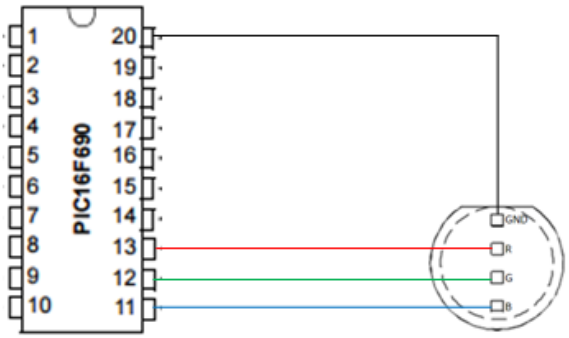


Figura 7.14. Connexions entre el LED RGB i el microcontrolador, per al circuit 2

7.2.5. LCD

El dispositiu LCD té 16 pins. A la *Taula 7.5* es mostra la funció de cada pin i, també, amb quin pin del microcontrolador es connecten. A l'esquema de la *Figura 7.15* es poden observar les connexions esmentades. Com es pot veure a l'esquema, el pin 3 de la LCD, el Vee, no va connectat al microcontrolador. Això es deu a què aquest pin serveix per modificar el contrast de la pantalla, que es controla mitjançant una resistència variable. Per altra banda, els pins del 7 al 10 tampoc es connecten al microcontrolador, ja que al circuit només s'utilitzen els pins D4 D5, D6 i D7 per rebre informació. El pin 16 del dispositiu LCD, el càtode del LED de la pantalla, es connecta al microcontrolador per mitjà d'una resistència per evitar que aquest LED es cremi.

Pin LCD	Funció	Pin microcontrolador
1	V_{ss}	20
2	V_{dd}	1
3	V_{ee}	-
4	RS	15
5	R/W	7
6	E	14
7	D0	-
8	D1	-
9	D2	-
10	D3	-
11	D4	6
12	D5	5
13	D6	8
14	D7	9
15	A	1
16	K	20

Taula 7.5. Relació pins LCD – microcontrolador

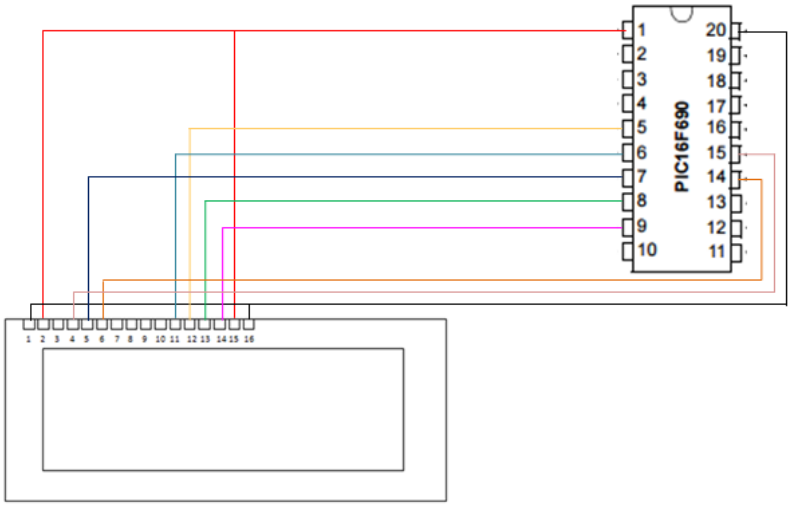


Figura 7.15. Connexions entre el dispositiu LCD i el microcontrolador

7.2.6. Polsador

El polsador que s'ha utilitzat en aquest projecte té dos pins. A la *Figura 7.16* es mostra com va connectat amb el microcontrolador, un dels pins es connecta al pin 10 del microcontrolador i l'altre, al terra.

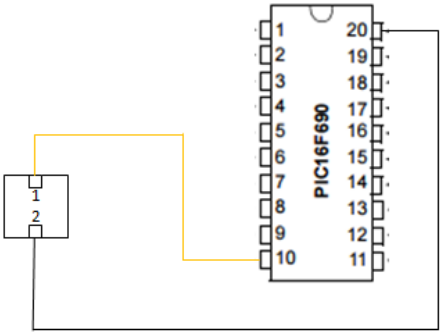


Figura 7.16. Connexions entre el polsador i el microcontrolador

7.2.7. Circuit 1: Connexió amb el dispositiu mòbil

El circuit que es fa servir per a la connexió del sistema amb el dispositiu mòbil consta de cinc elements: el PICkit2, el microcontrolador, el sensor de color, el mòdul *Bluetooth* i el LED RGB. Les interconnexions del circuit complet són les que es mostren a la *Figura 7.17.*

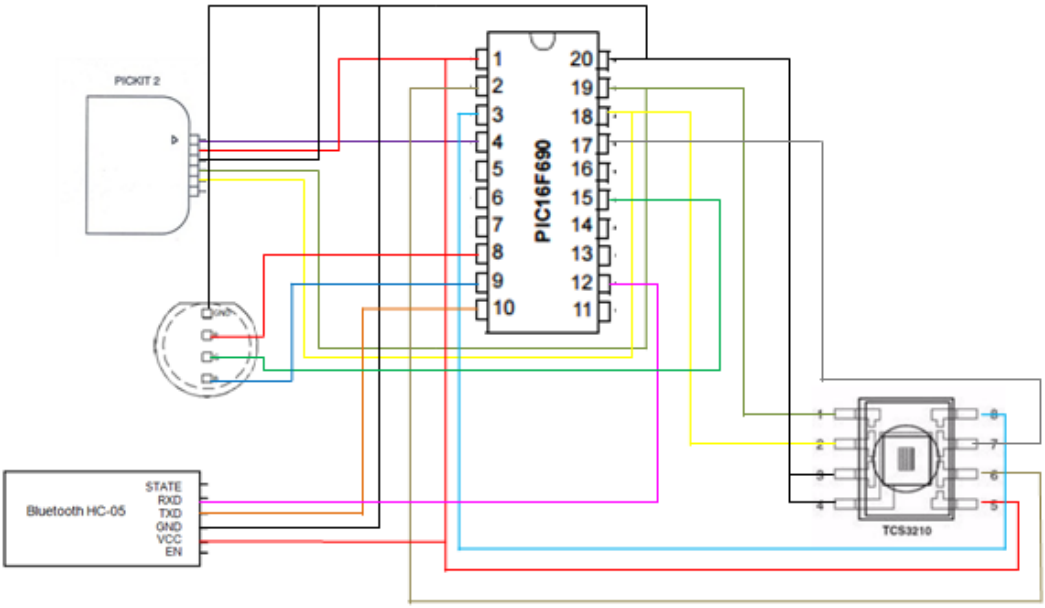


Figura 7.17. Connexions del circuit 1

7.2.8. Circuit 2: Connexió amb el dispositiu LCD

El circuit que intervé el dispositiu LCD consta de cinc elements: el microcontrolador, el sensor de color, la LCD , el polsador i el LED RGB, pel qual es mostra el color captat. A la figura següent, *Figura 7.18*, es poden veure les connexions entre aquests ells.

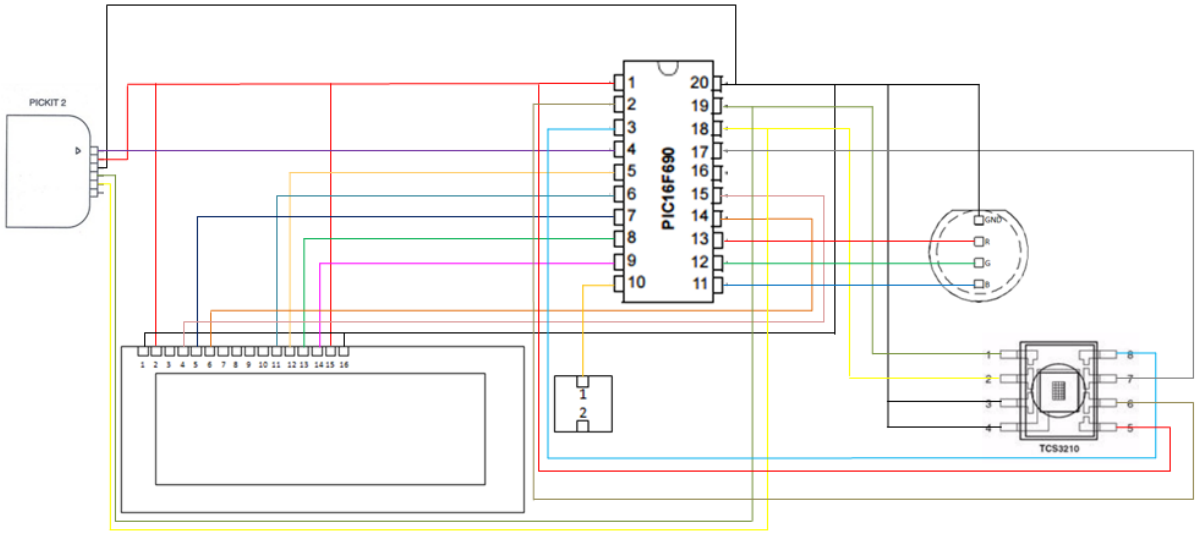


Figura 7.18. Connexions del circuit 2

7.3. Circuits finals

Durant el desenvolupament del projecte, el suport utilitzat per a dissenyar els circuits ha sigut una *protoboard* Aquesta, permet modificar fàcilment els components del circuit i les seves connexions. A la *Figura 7.19* es pot observar un exemple de circuit muntat sobre una *protoboard*.

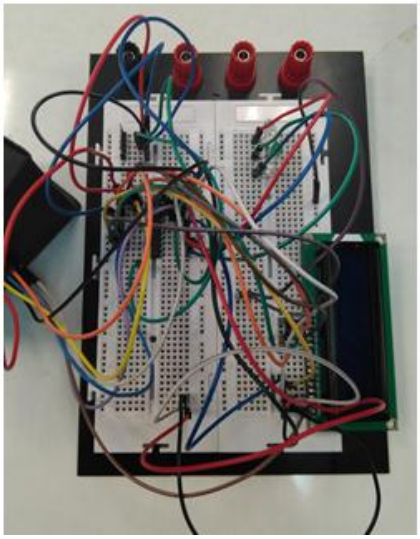


Figura 7.19. Circuit muntat sobre una protoboard

Un cop finalitzat el projecte, s'han soldat els circuits finals sobre d'una placa perforada. Els components no han estat soldats directament sobre la placa, sinó que s'han soldat uns connectors en el seu lloc. Els components es muntaran sobre aquests connectors. Això facilitarà el procés en cas que algun d'ells s'hagi de canviar. A la *Figura 7.20* i *a la Figura 7.21* es mostra el resultat final del circuit 1 i del circuit 2, respectivament.

A part de soldar les connexions explicades en l'apartat anterior, també s'hi ha soldat un regulador de tensió, que permet alimentar amb una pila de 9V el circuit, que en necessita menys.

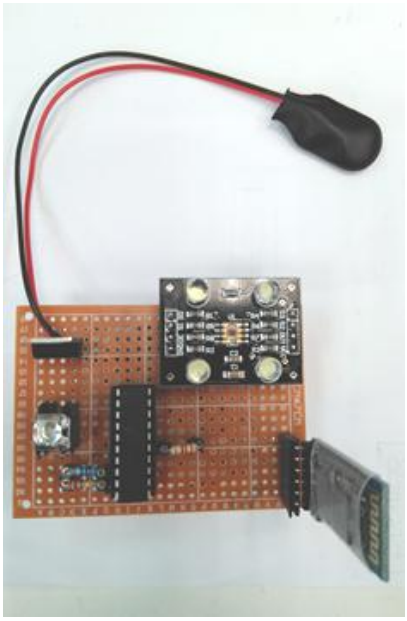


Figura 7.20. Circuit 1 (comunicació via Bluetooth amb dispositiu Android)

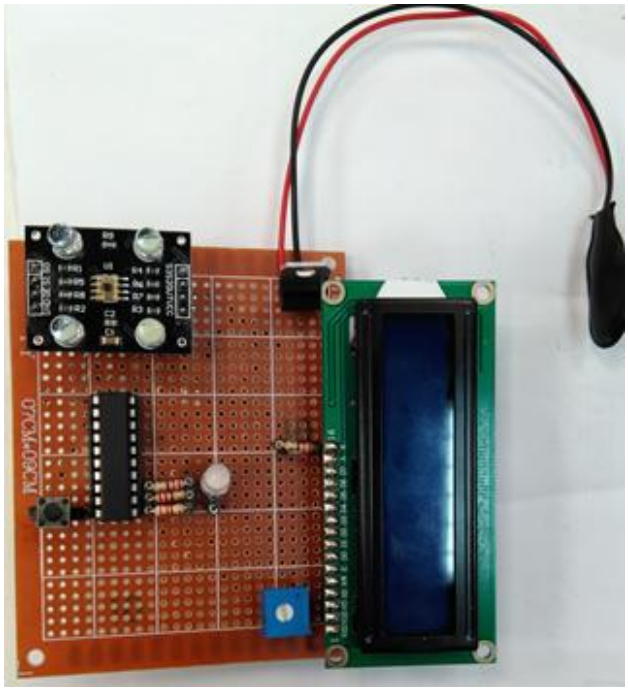


Figura 7.21. Circuit 2 (visualització de la información sobre una LCD)

8. Programació microcontrolador

En aquest capítol s'expliquen detalladament les funcions dels programes programats en el microcontrolador en ambdós circuits: el que es comunica amb el dispositiu Android i el que mostra la informació sobre una LCD. S'ha emprat el llenguatge C. A l'*Annex* es troben els programes sencers comentats. Hi ha una part del codi, com ara les inicialitzacions del microcontrolador i del sensor, que és comuna en els dos programes i només s'explica un cop.

8.1. Circuit 1: Connexió amb el dispositiu Android

Per aquest circuit, el programa té dos objectius. Per una banda, ha de poder rebre dades del telèfon mòbil i enviar-les al microcontrolador per tal de reproduir el color desitjat amb un LED RGB. Per l'altra banda, ha de poder enviar dades al telèfon mòbil per reproduir per la pantalla d'aquest el color captat pel sensor. Per tal de fer ambdues coses s'han creat dos arxius: Circuit_1.c i uart.h. En el primer hi consten les inicialitzacions necessàries del microcontrolador i del sensor, la funció principal, una funció auxiliar que mesura la freqüència dels colors captats pel sensor i el servei d'interrupcions. En el fitxer uart.h, hi consta la inicialització de l'UART i les funcions d'escriptura i de lectura d'aquest.

Abans d'explicar detalladament el codi del programa es mostra un diagrama de flux que pretén facilitar l'enteniment del funcionament d'aquest .

Tal com es pot veure en el *Diagrama 8.1, p*rimer es configura el microcontrolador i s'inicialitzen les variables. Tot seguit, es configuren els *timers* utilitzats (Timer1 i Timer0) i es configura l'UART. En aquest punt, s'espera rebre una dada, que l'enviarà l'usuari des del telèfon mòbil. Aquesta dada pot ser 0 o 1. Si la dada és 0, es mesura la freqüència de vermell, verd i blau que capta el sensor i s'escriuen aquests tres valors a l'UART per tal que puguin ser llegits des de la pantalla del telèfon mòbil. En canvi, si la dada és 1, es llegeixen els valors de l'UART (enviats des del telèfon mòbil) i s'encén el LED RGB amb les tres components rebudes. Tant en un cas com en l'altre, quan s'ha executat la funció es torna a esperar rebre una dada. Si no se'n rep cap, el programa no fa res més i res canvia sobre la pantalla del dispositiu mòbil. Quan es torna a rebre una dada, depenent de si es rep un 0 o un 1 es realitzarà un o altre procés.

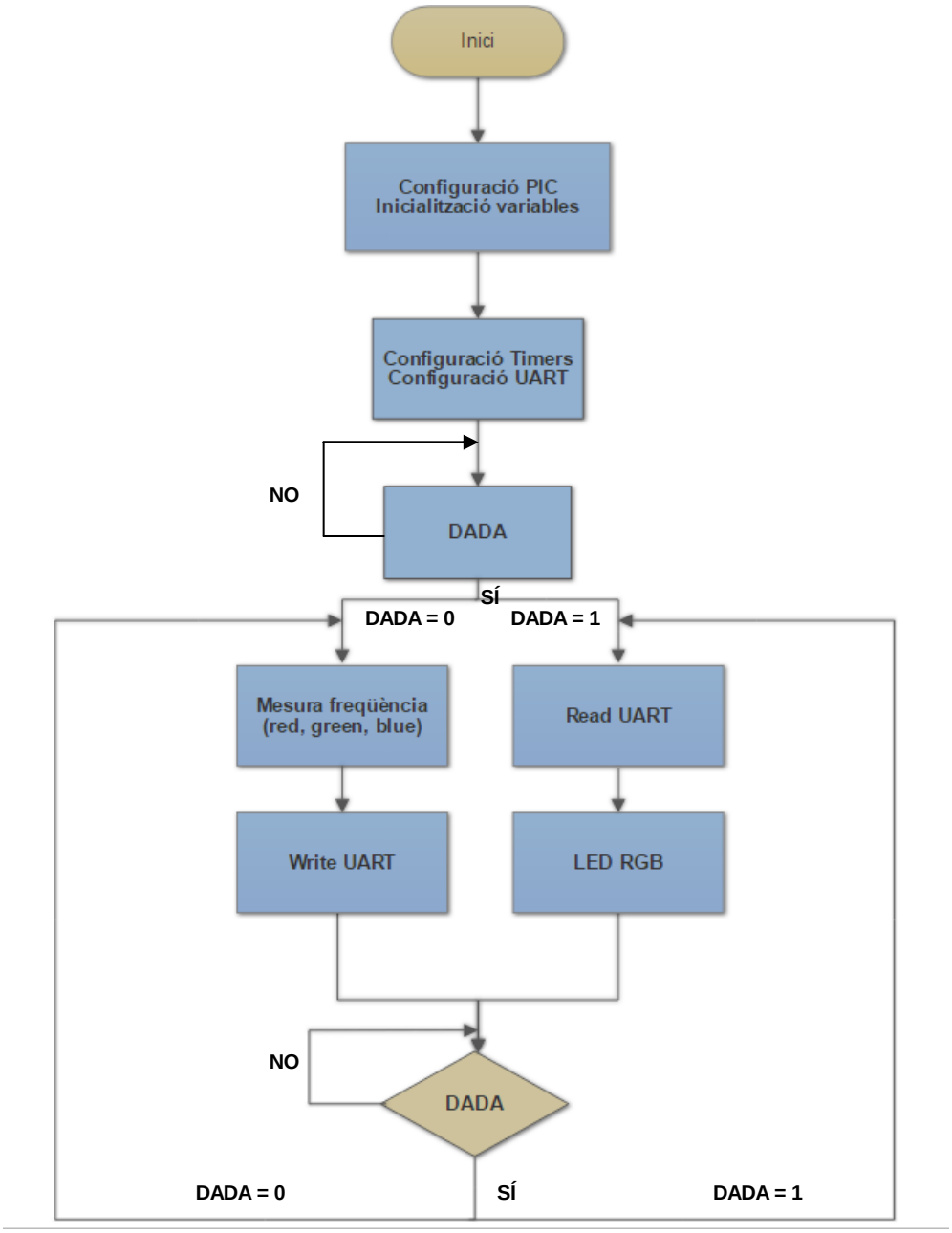


Diagrama 8.1. Diagrama de flux del programa del circuit 1

8.1.1. Circuit_1.C

CONFIGURACIONS I DEFINICIÓ DE VARIABLES

En el fitxer Circuit_1.c es comença carregant les llibreries necessàries: la del microcontrolador i la del compilador. A més, es defineix que la freqüència de funcionament del microcontrolador utilitzat és de 4MHz i s'inclou el fitxer uart.h. Per a la millor comprensió del programa, es defineix que ON = 1 i OFF = 0 (*Figura 8.1*).

```
// CONFIGURACIÓ GENERAL INICIAL
#include <pic16f690.h>  // S'inclou la llibreria del PIC16F690
#include <xc.h>     // S'inclou la llibreria del compilador XC
#include "uart.h"   // S'inclou el fitxer UART.h
#define _XTAL_FREQ 4000000  // Es defineix la freqüència de funcionament del microcontrolador
#define ON  1   // Es defineix que ON és 1
#define OFF 0   // Es defineix que OFF és 0
```

Figura 8.1. Configuració general inicial

A continuació, es configura el microcontrolador i es defineixen els pins que s'utilitzen pel sensor i pel led RGB (*Figura 8.2* i *Figura 8.3*, respectivament) amb l'ajuda dels seus respectius *Datasheet*. En la *Taula 8.1* s'indiquen quins són els bits que s'han configurat, amb quin valor s'ha fet i quina és la seva funció.

Bit	Valor	Funció
FOSC	INTRCIC	Selecciona rellotge intern de l'oscil·lador. Es fan disponibles els pins RA4 i RA5
WDTE	0	Watchdog Timer desactivat
PWRTE	0	Power-up Timer desactivat
MCLRE	0	Entrada digital
CP	0	Program memory code protection està activat
CPD	0	Data memory code protection està activat
BOREN	0	Brown-out reset bits desactivat
IESC	0	Internal External Switchover mode is disabled
FCMEN	0	Fail-Safe Clock Monitor està desactivat

Taula 8.1. Configuració inicial microcontrolador

```
// CONFIGURACIÓ MICROCONTROLADOR
#pragma config FOSC = INTRCIO // Es selecciona l'oscil·lador intern mitjançant els Oscillator Selection bits
                              // Disponibilitat dels pins RA4 i RA5
#pragma config WDTE = OFF     // Es desactiva el Watchdog Timer
#pragma config PWRTE = OFF    // Es desactiva el Power-up Timer
#pragma config MCLRE = OFF    // Entrada digital
#pragma config CP = OFF       // S'activa el program memory code protection
#pragma config CPD = OFF      // S'aciva el data memory code protection
#pragma config BOREN = OFF    // Es desactiven els brown-out selection bits. BOR desactivat
#pragma config IESO = OFF     // Es desactiva el mode Internal External Switchover
#pragma config FCMEN = OFF    // Es desactiva el Fail-Safe Clock Monitor
```

Figura 8.2. Configuració microcontrolador

Es defineixen que les sortides S0, S1, S2 i S3 del sensor seran els pins RA0, RA1, RA2, RA4 del microcontrolador, respectivament. S0 i S1 serveixen per escalar la freqüència de sortida i S2 i S3 per seleccionar el díode (Vermell, verd, blau o clear) amb el qual es mesura la freqüència. El senyal de sortida del sensor serà el pin RA5. Pel que fa al LED RGB, se li assignen els pins del PORTC RC6,RC1 i RC7.

```
// INFORMACIÓ DEL SENSOR
#define S0 RA0 // Es defineix que S0 és el pin RA0 del microcontrolador
#define S1 RA1 // Es defineix que S1 és el pin RA1 del microcontrolador
#define S2 RA2 // Es defineix que S2 és el pin RA2 del microcontrolador
#define S3 RA4 // Es defineix que S3 és el pin RA4 del microcontrolador
// OUT al RA5 que és el senyal del qual es vol mesurar la seva freqüència (rellotge extern TMR1 al pin RA5)

//INFORMACIÓ LED RGB
#define LEDR RC6 // Es defineix que LEDR és el pin RC6
#define LEDG RC1 // Es defineix que LEDG és el pin RC1
#define LEDB RC7 // Es defineix que LEDB és el pin RC7
```

Figura 8.3. Informació del sensor i del LED RGB

Abans de procedir al programa principal (*main*), es defineixen i s'inicialitzen algunes variables (*Figura 8.4).* Les variables amplepolsR, amplepolsG, amplepolsB, compteR, compteG, compteB, DAD, colorR, colorG i colorB són d'un byte (char). Les variables FREQ, FREQ_R, FREQ_G, FREQ_B i FREQ_C són positives i de 16 bits (unsigned int) ja que es necessiten més de 8 bits per expressar els seus valors. També es defineix la funció mesura_frequencia, que serà cridada des del *main*.

```
//DEFINICIÓ I INICIALITZACIÓ DE VARIABLES
char amplepolsR; // Es crea la variable amplepolsR
char amplepolsG; // Es crea la variable amplepolsG
char amplepolsB; // Es crea la variable amplepolsB
char compteR = 0; // Es crea la variable compteR i s'inicialitza a 0
char compteG = 0; // Es crea la variable compteG i s'inicialitza a 0
char compteB = 0; // Es crea la variable compteB i s'inicialitza a 0
char colorR, colorG, colorB; //Es creen les variables colorR, colorG i colorB
unsigned int FREQ, FREQ_R, FREQ_G, FREQ_B, FREQ_C;  // Es creen les diferents variables FREQ (sempre positives)
int Valor_R, Valor_G, Valor_B, Valor_C; // Es creen les variables enteres Valor_R, Valor_G, Valor_B i Valor_C
char DADA = 0; // Es crea la variable DADA i s'inicialitza a 0

unsigned int mesura_frequencia(void); // Es defineix la funció mesura_frequencia
```

Figura 8.4. Definició variables i funció mesura_frequencia

FUNCIÓ MAIN

En la funció principal (*main*) es realitzen diverses accions: s'inicialitza el microcontrolador, es configura el Timer1 i l'UART, es mesura la freqüència captada pel sensor amb l'ajuda de la funció auxiliar mesura_frequencia i s'escriuen les dades a l'UART per tal d'enviar-les al dispositiu mòbil.

Per tal d'inicialitzar el microcontrolador, se seleccionen tots els pins del PORT A, PORTB i PORTC com a sortides excepte el pin RA5 del PORTA, que se selecciona com a entrada. També es configura per tal que totes les entrades siguin digitals.

Per tal de fer-ho, cal consultar el *Datasheet* del microcontrolador. En la taula *Taula 8.2*, es poden veure els pins del registre TRISA. Si els seus bits prenen el valor de 1, aleshores el pin corresponent del PORTA actua com a entrada. Si prenen el valor de 0, actuen com a sortida. En aquest cas, només es prendrà com a entrada el bit TRISA5, per tal que el pin RA5 del PORTA actuï com a entrada. Aquesta entrada serà el valor captat pel sensor i serà el que s'agafarà per calcular la freqüència.

U-0	U-0	R/W-1	R/W-1	R-1	R/W-1	R/W-1	R/W-1
—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
bit 7							bit 0

Taula 8.2. Bits del registre TRISA del PORTA

Els registres ANSEL i ANSELH serveixen per configurar les entrades com analògiques si fos necessari. Caldrà posar-los tots a zero per tal de tenir sortides digitals. Els pins d'ambdós registres són els de la *Taula 8.3* i *Taula 8.4*, respectivament

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

Taula 8.3. Bits del registre ANSEL

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	ANS11	ANS10	ANS9	ANS8
bit 7							bit 0

Taula 8.4. Bits del registre ANSELH

Per tal de fer que tots els pins del PORTB i PORTC siguin sortides, anàlogament al que s'ha fet amb el PORTA, caldrà indicar que tots els bits dels registres TRISB i TRISC siguin 0. Els bits d'ambdós registres són els de la *Taula 8.5*.

R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0
TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—
bit 7							bit 0

R/W-1	R/W-1	R/W-1	R/W-1	R-1	R/W-1	R/W-1	R/W-1
TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
bit 7							bit 0

Taula 8.5.Bits dels registres TRISB i TRISC

En la *Figura 8.5*, es pot observar la inicialització dels registres comentats.

```
// Inicialització microcontrolador
TRISA = 0B00100000;; // Selecció de pins port A com a sortides excepte el pin RA5 que serà entrada
TRISC = 0B00000000;  // Selecció pins port B com a sortides
TRISB = 0x00;  // Selecció pins port B com a sortides
ANSEL = 0X00;  // Selecció sortides digitals
ANSELH = 0X00; // Selecció sortides digitals
```

Figura 8.5. Inicialització microcontrolador

Per configurar el Timer1, que és el perifèric que es fa servir per la funció mesura_frequencia, caldrà configurar el registre T1CON. Els seus bits són els de la *Taula 8.6.*

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T1GINV(1)	TMR1GE(2)	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Taula 8.6. Bits del registre T1CON

El bit TMR1GE (*Timer1 Gate Enable bit*) és un dels bits que es necessita configurar per habilitar el Timer1 i la seva funció depèn del TMR1ON (*Timer1 On bit*), que habilita o inhabilita en un nivell de lògica posterior el Timer1. Si TMR1ON és 0, el valor de TMR1GE no importa. En canvi, si el TMR1ON és 1 i el TMR1GE és 0, el Timer1 estarà activat. Per altra banda, si ambdós bits són 1, el Timer1 estarà actiu sempre que la seva porta estigui inactiva. Per inicialitzar-ho (*Figura 8.6*), s'ha posat 1 al bit TMR1GE i 0 al bit TMR1ON. És a dir, s'ha habilitat l'*Enable* Timer1 però el rellotge està desactivat (s'activarà quan escaigui). A més, s'ha activat el rellotge extern del Timer1 amb el bit TMR1CS (*Timer1 Clock Source Select bit*).

Per últim, s'ha activat el bit T1GSS (*Timer1 Gate Source Select bit*) del registre CM2CON1 del Timer1. Els bits d'aquest registre són els de la *Taula 8.7*. Posant aquest bit a 0 permet utilitzar el pin RA5/T1CKI com a entrada de rellotge del Timer1 en comptes d'utilitzar el rellotge intern del microcontrolador.

R-0	R-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0
MC1OUT	MC2OUT	—	—	—	—	T1GSS	C2SYNC
bit 7							bit 0

Taula 8.7. Bits del registre CM2CON1

A la *Figura 8.6* es poden veure totes les inicialitzacions del Timer1.

```
// CONFIGURACIÓ TMR1 PER mesura_frequencia
T1CON = 0B01000010;  // S'habiliten el Timer1 Gate Enable bit (TMR1GE) i el rellotge extern del Timer1 (TMR1CS)
T1CONbits.TMR1ON = OFF; // Timer1 desconnectat
CM2CON1bits.T1GSS = 0;  // Es desactiva el clock intern del TMR1 per a poder utilitzar RA4 com a entrada
```

Figura 8.6. Configuració TMR1

Un cop s'ha configurat el Timer1 es configura el Timer0 (*Figura 8.7)*, que és el que es fa servir en el servei d'interrupcions per a la funció que genera el PWM. S'inhabiliten les interrupcions del Timer0 amb el registre INTCON (*Taula 8.8*) posant el bit T0IE (*Timer0 Overflow Interrupt Enable bit*) a 0. A més, es posa OPTION_REG a 0 per tal de seleccionar el rellotge intern del Timer0. Els bits del registre OPTION_REG són els que es mostren en la *Taula 8.9*. El bit que serveix per seleccionar el rellotge intern del Timer0 és el T0CS (*Timer0 Clock Source Selection bit)*, però es posa tot el registre a 0 perquè tampoc es necessiten les funcions dels altres bits d'aquest registre.

```
//CONFIGURACIÓ TIMER0
INTCONbits.T0IE = 0;  // Es desactiven les interrupcions del Timer0
OPTION_REG = 0;       // Selecció rellotge intern pel Timer0
```

Figura 8.7. Configuració Timer0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RABIE(1,3)	T0IF(2)	INTF	RABIF
bit 7							bit 0

Taula 8.8. Bits del registre INTCON

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RABPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Taula 8.9. Bits del registre OPTION_REG

A continuació, es configura l'UART. Primer, es configuren els pins de RX i de TX com a entrada i sortida, respectivament, i es crida a la funció d'inicialització de l'UART indicant que la velocitats de transmissió seran 9600 bauds. Aquesta funció es troba en el fitxer uart.h i

està explicada més endavant. A més, també s'han activat les interrupcions globals i dels perifèrics i s'han habilitat els *enables* d'interrupció.

El registre INTCON és el que controla les interrupcions. Els bits d'aquest registre són els de la *Taula 8.8*. Per tal d'activar les interrupcions globals cal que GIE (*Global Interrupt Enable bit*) sigui 1 i per activar les interrupcions dels perifèrics cal que PEIE (*Peripherical Interrupt Enable bit*) sigui 1.

A més, per tal d'habilitar les interrupcions dels perifèrics, cal configurar alguns bits del registre PIE1 (*Peripherical Interrupt Enable Register*). Els bits d'aquest registre són els que es troben a la *Taula 8.10.*

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE(5)	RCIE(3)	TXIE(3)	SSPIE(4)	CCP1IE(2)	TMR2IE(1)	TMR1IE
bit 7							bit 0

Taula 8.10. Bits del registre PIE1

Posant a 1 els bits RCIE (*EUSART Receive Interrupt Enable bit)* I TXIE (*EUSART Transmit Interrupt Enable bit*), s'habiliten les banderes d'interrupció de recepció i de transmissió, respectivament.

En la *Figura 8.8* es pot veure la inicialització completa de l'UART.

```
// CONFIGURACIÓ UART
TRISBbits.TRISB7 = 0; // Es configura el pin RB7 com a sortida. Serà la transmissió de l'UART: TX
TRISBbits.TRISB5 = 1; // Es configura el pin RB5 com a entrada. Serà la recepció de l'UART: RX
UART_Init(9600);      // Es crida a la funció d'inicialització de l'UART (fitxer uart.h)
INTCONbits.PEIE = 1;  // S'activen les interrupcions dels perifèrics
INTCONbits.GIE = 1;   // S'activen les interrupcions globals
PIE1bits.RCIE = 1;    // S'habilita bandera interrupció recepció
PIE1bits.TXIE = 0;    // S'habilita bandera interrupció transmissió
```

Figura 8.8. Configuració UART

Tot seguit, tal com es pot veure en la *Figura 8.9*, s'inicialitza la variable FREQ a 0 i es configura el sensor per tal que l'escala de la freqüència de sortida sigui del 2%. Tal com es pot veure en la *Taula 8.11* on s'indiquen els valors que han de prendre S0 i S1 en funció de l'escala de freqüència de sortida desitjada, per a treballar amb una escala del 2% cal que S0 valgui 0 i S1 valgui 1.

S0	S1	OUTPUT FREQUENCY SCALING (f_o)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

Taula 8.11. Valors de S0 i S1 en funció de l'escala de la freqüència de sortida

```
FREQ = 0; // S'inicialitza la variable FREQ a 0

// S'escull una escala de freqüència del 2% a la sortida del sensor
S0 = 0;
S1 = 1;
```

Figura 8.9. Configuració escala freqüència de sortida del sensor

Un cop fetes totes les inicialitzacions i configuracions necessàries, el programa entra en el bucle idefinit *while (1)*. És a dir, s'executarà mentre el contingut del parèntesi sigui 1 (sempre). Quan es produeixi una interrupció, se sortirà d'aquest bucle i s'executarà el servei d'interrupcions.

Quan des del telèfon mòbil s'envia una dada, es produeix una interrupció. La variable DADA pren el valor del byte enviat. Si DADA és 0, aleshores s'executa la condició d'*if* de dins del *main*. El servei d'interrupcions està explicat amb detall més endavant.

A continuació, s'explica la condició *d'if* que s'executa dins del *main* (*Figura 8.10).* Es mesuren les freqüències captades pel sensor per cada un dels filtres: red, green, blue i clear i, quan l'usuari des del dispositiu mòbil demana els valors, aquests s'escriuen a la UART i es transmeten via *Bluetooth* fins al dispositiu.

Per tal de mesurar la freqüència captada pel sensor, primer es tria el filtre amb que es vol mesurar i després es crida a la funció mesura_frequencia (explicada més endavant). Les freqüències de cada filtre es guarden en les variables FREQ_R, FREQ_G, FREQ_B i FREQ_C. Tot seguit, es multipliquen per quatre i pel factor de conversió explicat en el capítol 5.2 per tal d'obtenir un número entre el 0 i el 255. D'aquesta manera, s'obtenen les components RGB del color mesurat. Aquest valor s'emmagatzema en les variables Valor_R, Valor_G i Valor_B.

A la *Taula 8.12*, es poden veure els valors que han de prendre S2 i S3 segons el filtre amb el que es vulgui mesurar.

S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

Taula 8.12. Valors de S2 i S3 segons el filtre del sensor

A continuació, s'activa la transmissió i s'escriuen els valors de les variables Valor_R, Valor_G i Valor_B a l'UART. Aquests valors són els que es mostren per la pantalla del telèfon mòbil. Abans d'escriure'ls a l'UART cal convertir-los a variables de 8 bits fent char(Valor_R), char(Valor_G) i char(Valor_B), respectivament. Entre l'escriptura d'una variable i la següent s'espera 2ms per tal que no se saturi el sistema. Un cop escrites les tres dades, s'atura la transmissió i el valor DADA passa a valer 2 per tal que no es torni a entrar a la condició *if* fins que l'usuari no ho torni a demanar.

El codi de tot aquest procés és el que es mostra en la *Figura 8.10.*

```
if (DADA==0) // Si s'envia un 0 des de l'app, es llegeixen els valor captats pel sensor i s'envien els tres valors RGB al telèfon
{

amplepolsR = 0; //Fa que s'apagui el vermell
amplepolsG = 0; //Fa que s'apagui el verd
amplepolsB = 0; //Fa que s'apagui el blau

// RED (vermell)
S2 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre vermell (RED)
S3 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre vermell (RED)
FREQ_R = mesura_frequencia(); // La variable FREQ_R pren el valor del resultat de la funció mesura_frequencia
Valor_R=(4*FREQ_R)*(255/1316.); // La variable Valor_R pren el valor de la freqüència en vermell
                                // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255

// GREEN (verd)
S2 = ON; // Configuració per tal que es mostri la frequencia mesurada segons el filtre verd(GREEN)
S3 = ON; // Configuració per tal que es mostri la frequencia mesurada segons el filtre verd(GREEN)
FREQ_G = mesura_frequencia(); // La variable FREQ_G pren el valor del resultat de la funció mesura_frequencia
Valor_G=(4*FREQ_G)*(255/1316.); // La variable Valor_G pren el valor de la freqüència en verd
                                // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255

// BLUE (blau)
S2 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre blau (BLUE)
S3 = ON;  // Configuració per tal que es mostri la frequencia mesurada segons el filtre blau (BLUE)
FREQ_B = mesura_frequencia(); // La variable FREQ_B pren el valor del resultat de la funció mesura_frequencia
Valor_B=(4*FREQ_B)*(255/1316.); // La variable Valor_B pren el valor de la freqüència en blau
                                // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255

// CLEAR (blanc)
S2 = ON;  // Configuració per tal que es mostri la frequencia mesurada segons el filtre blanc (CLEAR)
S3 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre blanc (CLEAR)
FREQ_C = mesura_frequencia(); // La variable FREQ_C pren el valor del resultat de la funció mesura_frequencia
Valor_C=(4*FREQ_C)*(255/1316.); // La variable Valor_C pren el valor de la freqüència en blanc
                                // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255


TXEN=1; // S'activa la transmissió
UART_Write((char)Valor_R); // Es transforma Valor_R en una variable de 8 bits i s'escriu a l'UART
__delay_ms(2); // S'esperen 2ms
UART_Write((char)Valor_G); // Es transforma Valor_G en una variable de 8 bits i s'escriu a l'UART
__delay_ms(2); // S'esperen 2ms
UART_Write((char)Valor_B); // Es transforma Valor_B en una variable de 8 bits i s'escriu a l'UART
__delay_ms(2); // S'esperen 2ms
TXEN=0; // S'atura la transmissió perquè només cal enviar les dades un cop
DADA=2; // La variable DADA passa a valdre 2.
        // Així, no torna a entrar en aquest if fins que no es torni a enviar un 0 des del mòbil

}//Final de l'if
```

Figura 8.10. Condició if quan DADA val 0

FUNCIÓ MESURA_FREQUENCIA

Quan el programa principal està executant el bucle *while*, crida a la funció mesura_frequencia per cada un dels filtres. Aquesta funció (*Figura 8.11)*, activa el Timer1 posant el bit TMR1ON a 1 i l'inicialitza a 0. . El rellotge que activa el Timer1 es correspon amb el senyal del sensor del qual volem saber la freqüència. S'espera un quart de segon, llegeix quin valor té i l'emmagatzema a la variable FREQ. Finalment, desactiva el Timer1 i retorna el valor mesurat.

```
//FUNCIÓ QUE MESURA LA FREQÜÈNCIA R,G,B I CLEAR
unsigned int mesura_frequencia(void)  //Aquesta funció capta la informació del sensor
    {
    T1CONbits.TMR1ON = 1; // S'activa el Timer1
    TMR1 = 0;             // Es posa el Timer1 a 0
    __delay_ms(250);      // S'espera un quart de segon
    FREQ = TMR1;          // Es llegeix el valor del Timer1 i es guarda a la variable FREQ
    T1CONbits.TMR1ON = 0; // Es desactiva el Timer1
    return(FREQ);         // Retorna el valor de la variable FREQ (Timer1)
```

Figura 8.11. Funció mesura_frequencia

SERVEI D'INTERRUPCIONS

Quan arribi una paraula al RCREG Register de l'UART, s'activarà la bandera de recepció del buffer d'entrada de l'UART i s'executarà el servei d'interrupcions.

El registre PIR1 (*Peripherical Interrupt Request Register)* té els bits que es mostren en la *Taula 8.13*. Quan hi ha una paraula per llegir, el bit RCIF (*EUSART Interrupt Flag bit*) passa a valer 1 i la dada rebuda s'obté llegint el registre RCREG.

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF(5)	RCIF(3)	TXIF(3)	SSPIF(4)	CCP1IF(2)	TMR2IF(1)	TMR1IF
bit 7							bit 0

Taula 8.13. Bits del registre PIR1

Tal com es pot veure a la *Figura 8.12*, quan el bit RCIF del registre PIR1 sigui 1, s'executaran les comandes de dins del *if*.

El primer valor que es llegeix s'emmagatzema a la variable DADA. Si aquest valor és 0, s'executa la condició d'*if* del *main* explicada prèviament. En canvi, si el valor enviat és 1, s'executa la condició *if* de la *Figura 8.14.* Des de l'App, s'envia un 1 quan es volen enviar dades del color generat per l'usuari.

Les variables colorR, colorG i colorB, passen a valer els valors de la funció UART_Read() (funció del fitxer uart.h, explicada més endavant). Aquestes variables són les components RGB del color generat per l'usuari des de l'App i tenen un valor comprès entre el 0 i el 255. Per tal d'obtenir un valor entre el 0 i el 100, es multipliquen per 100 i es divideixen entre 255. Aquest valor s'emmagatzema en les variables amplepolsR, amplepolsG i amplepolsB. Un cop acabada la lectura, DADA passa a valer 2 per tal que no es torni a entrar a la condició if fins que l'usuari no ho torni a demanar. Es posa el TMR0 a 215. Aquest, s'inicialitza a 215 ja que si s'inicialitzés a un valor més baix, el període seria massa alt i, per tant, la freqüència seria molt baixa i es veuria el LED fent pampallugues. Cal tenir en compte que l'ull humà només percep llum estable a partir *dels 50 Hz aproximadament.* Per últim, s'activen les interrupcions del TMR0.

```
if (PIR1bits.RCIF) //Sempre que la bandera de recepció en el buffer d'entrada de l'UART estigui activada
{
DADA = UART_Read(); // Es llegeix el primer valor enviat des del mòbil.
                    // Si dada és 0, anirà a executar l'if de la funció main. Sinó, s'executarà l'if de les interrupcions
    if (DADA == 1) // Si s'envia un 1 des de l'app, es llegeixen les tres variables RGB enviades a continuació pel mòbil
    {
    colorR = UART_Read();  // Llegeix la dada en RX de l'UART i s'associa a la variable amplepolsR (serà un número entre el 0 i el 255)
    colorG = UART_Read();  // Llegeix la dada en RX de l'UART i s'associa a la variable amplepolsG (serà un número entre el 0 i el 255)
    colorB = UART_Read();  // Llegeix la dada en RX de l'UART i s'associa a la variable amplepolsB (serà un número entre el 0 i el 255)
    amplepolsR = (colorR*100)/255; // Es passa el valor de colorR entre un número del 0 al 100
    amplepolsG = (colorG*100)/255; // Es passa el valor de colorG entre un número del 0 al 100
    amplepolsB = (colorB*100)/255; // Es passa el valor de colorB entre un número del 0 al 100
    DADA = 2;// La variable DADA passa a valdre 2.
             // Així, no torna a entrar en aquest if fins que no es torni a enviar un 1
    TMR0 = 215;
    INTCONbits.T0IE = 1; // S'activen les interrupcions del Timer0
    }
```

Figura 8.12. Lectura de les dades enviades a l'UART des del dispositiu mòbil

Per últim, quan s'activa la bandera d'interrupció del Timer0, el bit T0IF (*Timer0 Overflow Interrupt Flag)* del registre INTCON es posa a 1 (*Taula 8.8)* indicant que ha transcorregut la unitat de temps prevista pel Timer0 (40 polsos de rellotge intern). Aleshores, s'executen les comandes de la *Figura 8.13*. La funció d'aquest *if* és modular l'amplitud del pols de l'ona de cada filtre (red, green i blue). Els comptadors compteR, compteG i compteB incrementen una unitat cada cop que s'entra al bucle. Quan el comptador arriba a l'amplitud del pols + 1, el LED s'apaga. D'aquesta manera, es pot modular el temps que està encès cadascun dels colors. El fet que se li sumi un a l'amplitud del pols és degut a que com que la primera vegada que entra al bucle ja se li suma 1, si l'amplitud del pols fos 0, el LED mai s'apagaria.

Quan la variable compteR arriba a 100, els tres LEDs s'encenen i es torna a començar el procés. S'agafa la variable compteR, però totes tres arribaran alhora al 100 ja que totes van incrementant a la vegada. El Timer0 s'inicialitza a 215.

```
    if (INTCONbits.T0IF)  // Sempre que la bandera de recepció del buffer d'entrada del Timer0 estigui activada
    {
        compteR++;  // S'incrementa una unitat cada cop que entra en el bucle
        compteG++;  // S'incrementa una unitat cada cop que entra en el bucle
        compteB++;  // S'incrementa una unitat cada cop que entra en el bucle
        if (compteR==amplepolsR + 1) //Quan l'incrementador compteR sigui igual a amplepolsR + 1
        {
            LEDR = 0; // La variable LEDR pren valor 0 (LED vermell s'apaga)
        }
        if (compteG==amplepolsG + 1) // Quan l'incrementador compteG sigui igual a amplepolsG + 1
        {
            LEDG = 0; // La variable LEDG pren valor 0 (LED verd s'apaga)
        }
        if (compteB==amplepolsB + 1) // Quan l'incrementador compteB sigui igual a amplepolsB + 1
        {
            LEDB = 0; //La variable LEDB pren valor 0 (LED blau s'apaga)
        }
        if (compteR == 100) // Quan l'incrementador arriba a 100
        {
            LEDR = 1; // La compoment vermella del LED RGB s'encén
            LEDG = 1; // La compoment verda del LED RGB s'encén
            LEDB = 1; // La compoment blava del LED RGB s'encén
            compteR = 0; // La variable compteR pren valor 0
            compteG = 0; // La variable compteG pren valor 0
            compteB = 0; // La variable compteB pren valor 0
        }
        TMR0 = 215; // El Timer0 es posa a 215, un valor prou alt perquè no es vegin pampallugues al LED RGB
        INTCONbits.T0IF = 0;  // Es desactiva la bandera d'interrupció del Timer0
    }
} //Final servei interrupcions
```

Figura 8.13. Funció que regula l'amplitud del pols de red, green i blue

8.1.2. UART.h

En aquest fitxer, s'inicialitzen alguns registres del mòdul UART i es defineixen les funcions de lectura i escriptura d'aquest.

Tal com ja s'ha prèviament definit al capítol 5.1.6, el registre SPBRG pren el valor 25. A més, també s'inicialitzen els registres BAUDCTL (*Baud Rate Control)*, RCSTA (*Receive Status and Control)* i TXSTA (*Transmit Status and Control)*. Els bits d'aquests registres són els de la *Taula 8.14*.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BAUDCTL	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D

Taula 8.14. Bits dels registres associats a l'UART

A la *Taula 8.15* es mostren els valors que s'han assignat als bits de cada registre i què s'està activant/desactivant amb aquesta inicialització. Es comenten els registres inicialitzats que són més importants. La resta de registres no esmentats, que són 0, no tenen cap funció en aquest projecte.

Del registre TXSTA s'inicialitzen els bits TXEN (*Transmit Enable bit*), SYNC (*EUSART Mode Select bit)* i BRGH (*High Baud Rate Select bit*). Del registre BAUDCTL s'inicialitza el bit BRG16 (*16-bit Baud Rate Generator bit*). Per últim, del registre RCSTA s'inicialitzen els bits SPEN (*Serial Port Enable bit*), SREN (*Single Receive Enable bit*) i CREN (*Continuous Receive Enable bit*).

Inicialització registres	Bits inicialitzats	Valor	Acció
TXSTA = 0x24	TXEN	1	Activa la transmissió
	SYNC	0	Selecció mode asíncron
	BRGH	1	Selecció alta velocitat
SPBRG = 25			Segons taula 12-5 del *Datasheet*
BAUDCTLbits.BRG16 = 0	BRG16	0	*Timer* 16 bits desactivat
RCSTA = 0xB0	SPEN	1	Configura els pins RX i TX com a ports en sèrie
	SREN	1	Valor *don't care* en mode asíncron
	CREN	1	Activa receptor

Taula 8.15. Taula resum inicialització registres UART

A la *Figura 8.14*, s'observen totes les inicialitzacions comentades.

```
//INICIALITZACIÓ UART
UART_Init(int baudrate)
{
    TXSTA=0x24; // S'activa la transmissió, se selecciona mode asíncron i se selecciona alta velocitat
    SPBRG=25;   // Se selecciona velocitat de transmissió
    BAUDCTLbits.BRG16=0; // Es desactiva timer 16 bits
    RCSTA=0xB0; // Es configuren els pins RX i TX com a ports en sèrie i s'activa mode recepció continua
}
```

Figura 8.14. Inicialització registres UART

A continuació, es defineixen les funcions de lectura i escriptura de l'UART. La funció d'escriptura es farà servir per escriure en l'UART les dades captades pel sensor i després transmetre-les al dispositiu mòbil, mentre que la funció de lectura es farà servir per llegir les dades enviades a través del mòbil.

TRMT (*Transmit Shift Register Status bit*), és un bit del register TXSTA (veure *Taula 8.14)* que indica si s'està efectuant una transmissió o no. Quan TRMT és 0, la transmissió està en curs. Quan és 1, la transmissió ja s'ha efectuat. En la funció UART_Write (*Figura 8.15*), mentre el TRMT sigui 0, el microcontrolador espera a escriure la dada al registre de transmissió de dades. Quan la transmissió estigui acabada, TRMT valdrà 1 i no s'escriurà

res més a l'UART.

```
//FUNCIÓ ESCRIPTURA
void UART_Write(char data)
{
 while(!TRMT); //Quan TRMT sigui 0, la transmissió està en curs
 TXREG = data; //Es guarda a TXREG la dada
}
```

Figura 8.15. Funció escriptura UART

RCIF (*EUSART Receive Interrupt Flag bit*), es un bit del registre PIR1 (veure *Taula 8.12*) que s'activa quan la recepció està completa. Mentre RCIF sigui 0, la recepció està en curs o no s'ha iniciat. Quan és 1, la recepció ja s'ha efectuat. En la funció UART_Read (*Figura 8.16*), mentre la recepció estigui en curs la bandera de recepció està desactivada (RCIF = 0) i es va llegint les dades. Quan la recepció s'ha finalitzat, s'activa la bandera, RCIF passa a ser 1 i el byte rebut és emmagatzemat en el registre RCREG.

```
//FUNCIÓ LECTURA
char UART_Read()
{
  while(!RCIF); // Mentre RCIF sigui 0 la bandera de recepció està desactivada i va llegint
  return RCREG; //Retorna el valor que llegeix
}
```

Figura 8.16. Funció de lectura de l'UART

8.2. Circuit 2: Connexió amb el dispositiu LCD

De la mateixa manera que amb el circuit 1, abans de procedir amb l'explicació del codi es presenta un diagrama de flux que resumeix el funcionament del programa d'aquest circuit.

Tal com es pot veure en el *Diagrama 8.2,* primer es configura el microcontrolador i s'inicialitzen les variables. A continuació, es configura la pantalla LCD i es defineixen les variables. Es mesuren les freqüències captades pel sensor i es mostren les components RGB del color captat. Mentre no es premi el polsador, es van llegint contínuament els valors captats pel sensor i es va actualitzant la pantalla. Quan es prem el polsador, s'atura la lectura i s'encén el LED RGB amb les components que hi ha en el moment de prémer el polsador. Mentre no es torni a prémer el polsador, el LED es manté encès. Quan es torna a prémer, el sensor comença de nou a mesurar les freqüències.

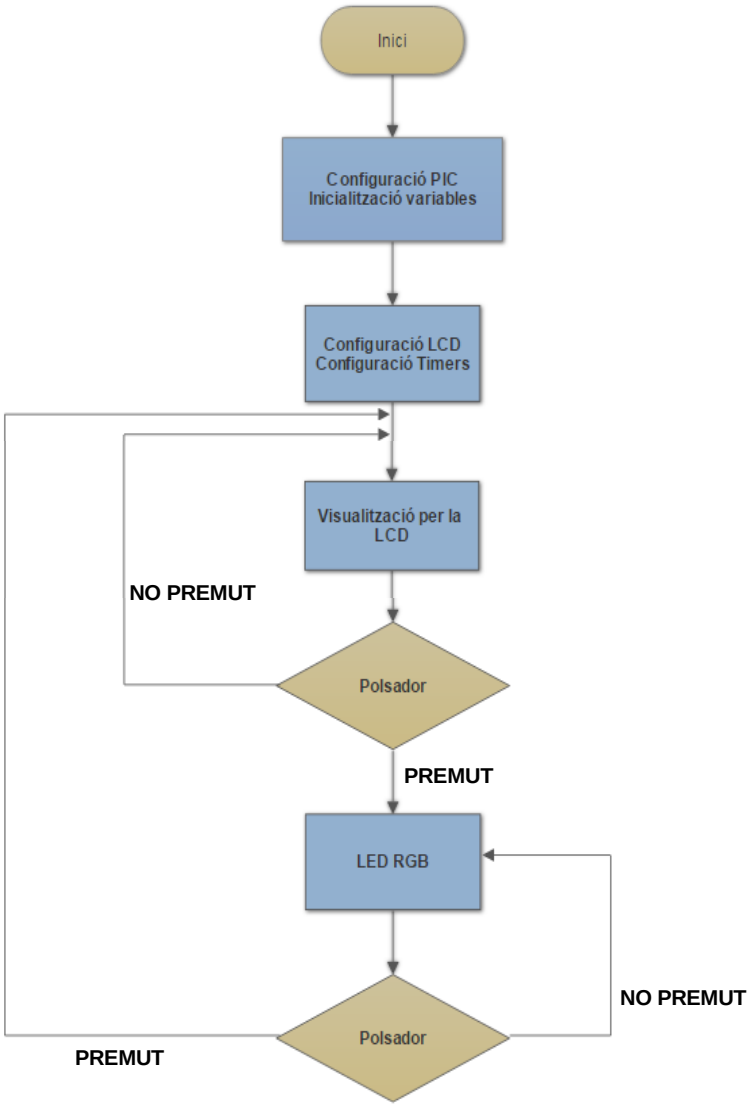


Diagrama 8.2. Diagrama de flux del programa del circuit 2

Com s'ha comentat en la introducció d'aquest capítol, la primera part de la programació, és comuna pels dos circuits, tant en el que intervé el dispositiu mòbil com en el de la pantalla LCD, ja que és la configuració general del circuit, on s'inclouen les llibreries bàsiques, i també la configuració del microcontrolador, usat en els dos circuits (*Figura 8.17*).

```
//PROGRAMACIÓ DEL MICROCONTROLADOR PER AL CIRCUIT 2: CONNEXIÓ AMB DISPOSITIU LCD

// CONFIGURACIÓ GENERAL INICIAL
#include <pic16f690.h> // Inclou la llibreria pic16f690.h, la del microcontrolador
#include <xc.h> // Inclou la llibreria xc.h, la del compilador
#define _XTAL_FREQ 4000000 //Defineix la freqüència de funcionament del microcontrolador, 4MHz
#define ON  1 // Es defineix que ON és 1
#define OFF 0 // Es defineix que OFF és 0

// CONFIGURACIÓ MICROCONTROLADOR
#pragma config FOSC = INTRCIO // Es selecciona l'oscil·lador intern mitjançant els Oscillator Selection bits
#pragma config WDTE = OFF // El Watchdog Timer Enable bit està desactivat
#pragma config PWRTE = OFF // El Power-up Timer Enable bit està activat
#pragma config MCLRE = OFF // El MCLR Pin Function Select bit és una entrada digital
#pragma config CP = OFF // El Code Protection bit està activat
#pragma config CPD = OFF // El Data Code Protection bit està activat
#pragma config BOREN = OFF // Els Brown-Out Reset bits estan desactivats
#pragma config IESO = OFF // El Internal Extrnal Switchover bit està desactivat
#pragma config FCMEN = OFF // El Fail-Safe Clock Monitor Enabled bit està desactivat
```

Figura 8.17. Configuració general i configuració del microcontrolador

A continuació, seguint amb les configuracions, es procedeix amb la configuració de la LCD, *Figura 8.18*, on es defineixen les connexions dels seus pins amb els pins del microcontrolador. A més, en aquesta part, també s'inclou la llibreria del model de LCD utilitzat. En aquesta llibreria s'inicialitzen les funcions pròpies del dispositiu LCD. Per altra banda, s'utilitza un altre fitxer per a poder programar la LCD correctament, aquest té el mateix nom que la llibreria però l'extensió és .c, ja que es tracta d'un fitxer principal i no d'una llibreria. En aquest programa exclusiu per la LCD i que serveix d'auxiliar del que es programa per aquest circuit, es programen les funcions pròpies del dispositiu, inicialitzades a la seva llibreria. Aquests dos fitxers, lcd_hd44780_pic16.c i lcd_44780_pic16.h, es troben a l'*Annex 2.1* i a l'*Annex 2.2*, respectivament.

```
// CONFIGURACIÓ PANTALLA LCD
#define RS RC1 // El RS, Register Select, de la LCD està connectat al pin RC1 del microcontrolador
               // quan s'escrigui RS es referirà a aquest pin
#define E RC2 // El E, Enable, de la LCD està connectat al pin RC2, quan s'escrigui E es referirà a aquest pin
#define RW RC3 // El RW, Read/Write, de la LCD està connectat al pin RC3, quan s'escrigui RW es referirà a aquest pin
#define D4 RC4 // El D4, Data4, de la LCD està connectat al pin RC4, quan s'escrigui D4 es referirà a aquest pin
#define D5 RC5 // El D5, Data5, de la LCD està connectat al pin RC5, quan s'escrigui D5 es referirà a aquest pin
#define D6 RC6 // El D6, Data6, de la LCD està connectat al pin RC6, quan s'escrigui D6 es referirà a aquest pin
#define D7 RC7 // El D7, Data7, de la LCD està connectat al pin RC7, quan s'escrigui D7 es referirà a aquest pin
#include "lcd_hd44780_pic16.h" // Inclou la llibreria lcd_hd44780_pic16.h, la del model de LCD utilitzat
```

Figura 8.18. Configuració del dispositiu LCD

També és comuna la part on es configuren el LED RGB i el sensor de color, tot i que poden ser diferents els pins del microcontrolador on es connecten, el procediment és el mateix en la programació d'ambdós circuits, *Figura 8.19*.

```
// CONFIGURACIÓ LED RGB
#define LEDR RB4 // El LEDR está connectat al pin RB4 del microcontrolador
                 // quan s'escrigui LEDR es referirà a aquest pin
#define LEDG RB5 // El LEDG está connectat al pin RB5, quan s'escrigui LEDG es referirà a aquest pin
#define LEDB RB6 // El LEDB está connectat al pin RB6, quan s'escrigui LEDB es referirà a aquest pin

// CONFIGURACIÓ SENSOR DE COLOR
#define S0 RA0 // El pin S0 del sensor està connectat al pin RA0 del microcontrolador
               // quan s'escrigui RA0 es referirà a aquest pin
#define S1 RA1 // El pin S1 del sensor està connectat al pin RA1 del microcontrolador
               // quan s'escrigui RA1 es referirà a aquest pin
#define S2 RA2 // El pin S2 del sensor està connectat al pin RA2 del microcontrolador
               // quan s'escrigui RA2 es referirà a aquest pin
#define S3 RA4 // El pin S3 del sensor està connectat al pin RA4 del microcontrolador
               // quan s'escrigui RA4 es referirà a aquest pin
// OUT al RA5 que és el senyal del qual es vol mesurar la seva freqüència (rellotge extern TMR1 al pin RA5)
```

Figura 8.19. Configuració del LED RGB i del sensor de color

Per acabar amb les configuracions, es realitza la del polsador, com es veu en la *Figura 8.20.*

```
// CONFIGURACIÓ DEL POLSADOR
#define boto RB7 // El polsador està connectat al pin RB7 del microcontrolador, quan s'escrigui boto es referirà a aquest pin
#define premut 0 // Es defineix que premut és 0
#define no_premut 1 // Es defineix que no_premut és 1
```

Figura 8.20. Configuració del polsador

Seguidament, com en el cas del circuit 1 (comunicació amb el dispositiu mòbil), es procedeix amb la definició i la inicialització de les variables necessàries utilitzades al llarg del programa i amb la creació de la funció *mesura_frequencia* (*Figura 8.21*), que servirà per captar la informació del sensor de color, i que serà cridada des de la funció *main*, la principal.

```
// DEFINICIÓ I INICIALITZACIÓ DE VARIABLES
char amplepolsR; // Es crea la variable amplepolsR
char amplepolsG; // Es crea la variable amplepolsG
char amplepolsB; // Es crea la variable amplepolsB
char compteR = 0; // Es crea la variable compteR i s'inicialitza a 0
char compteG = 0; // Es crea la variable compteG i s'inicialitza a 0
char compteB = 0; // Es crea la variable compteB i s'inicialitza a 0
bit estat = 0 ; // Es crea la variable binaria estat i s'inicialitza a 0
unsigned int FREQ, FREQ_R, FREQ_G, FREQ_B, FREQ_C; // Es creen les variables enteres i positives
                                                   // anomenades FREQ, FREQ_R, FREQ_G, FREQ_B i FREQ_C
int Valor_R, Valor_G, Valor_B, Valor_C; // Es creen les variables enteres Valor_R, Valor_G, Valor_B i Valor_C

unsigned int mesura_frequencia(void); // Es crea la funció mesura_frequencia
                                      // que és la que servirà per mesurar la freqüència dels colors
```

Figura 8.21. Creació de variables i de la funció mesura_frequencia

Seguint amb el programa, es comença amb la funció principal, la funció *main*. Com s'ha explicat en l'apartat anterior, aquesta funció realitza varies accions, entre les quals està inicialitzar el microcontrolador, es mesura la dels colors captada pel sensor, mitjançant la funció *mesura_frequencia*, on s'utilitza el *Timer1* que, per tant, també s'ha configurat i inicialitzat en aquesta funció. En la figura següent, *Figura 8.22*, es veu la part del programa

on s'ha inicialitzat el microcontrolador i el *Timer1*, l'explicació de perquè s'ha fet com en el programa està a l'apartat anterior, en el del circuit 1, el del dispositiu mòbil.

```
// FUNCIÓ PRINCIPAL
void main (void)
{
    // INICIALITZACIÓ DEL MICROCONTROLADOR
    TRISA = 0B00100000; // Configurar el PORTA com a sortida, excepte el pin RA5
    TRISC = 0B00000000; // Configurar el PORTC com a sortida
    TRISB = 0x80; // Configurar el pin RB7 com a entrada, els altres del PORTB com a sortida
    ANSEL = 0X00; // Configuració de mode digital
    ANSELH = 0X00; // Configuració de mode digital
    OPTION_REG = 0B00000000; // S'activa el Pull-up general del PORT AB
                             // per definir que el polsador, quan no està premut, està a 1 i, aquest, està connectat al pin RB7
    WPUB = 0B10000000; // S'activa el Weak Pull-up del pin RB7
                       // necessari pel funcionament del polsador, que es troba connectat en aquest pin

    // CONFIGURACIÓ TMR1 PER MESURAR LA FREQÜÈNCIA
    T1CON = 0B01000010; // S'habiliten el Timer1 Gate Enable bit, TMR1GE, i el rellotge extern del Timer1, TMR1CS.
                        // A més, aquest pin és el RA5, que és on va connectada la sortida del sensor, la freqüència
    T1CONbits.TMR1ON = OFF; // Inicialment, el Timer1 està desconnectat
    CM2CON1bits.T1GSS = 0; // El Timer1 Gate Source Select, T1GSS, es posa a 0 per a poder utilitzar el pin RA4 com a entrada.
                           // Si es posés a 1, aquest pin seria el pin de clock del Timer1
```

Figura 8.22. Inicialització del microcontrolador i configuració del Timer1

Referent a la inicialització del microcontrolador, a l'apartat anterior no estan explicats els registres *OPTION_REG* ni *WPUB*, ja que tenen a veure amb el polsador i en l'altre circuit no n'hi ha cap. Per a poder configurar bé aquests registres, és necessari consultar el *Datasheet* del microcontrolador. Pel que fa al *OPTION_REG*, *Taula 8.16*, tots els seus bits es posen a 0, per desactivar-los tots excepte el $\overline{\text{RABPU}}$ (*pull-up*), que està negat i, per tant, amb un 0 activa els *pull-ups* del PORTA i el PORTB, on hi ha el polsador .

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RABPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Taula 8.16. Bits del registre OPTION_REG

Pel que fa al *WPUB*, *Taula 8.17*, tots els seus bits es posen a 0 excepte el bit 7, que es posa a 1, per activar el *pull-up* del B7, *WPUB7*, pin on està connectat el polsador. És necessari activar-lo per determinar que, quan no està premut, el polsador es troba a 1.

R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0
WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—
bit 7							bit 0

Taula 8.17. Bits del registre WPUB

A la funció *main* també s'han d'habilitar les interrupcions dels perifèrics i les globals i les interrupcions del *Timer0* comencen desconnectades, *Figura 8.23*. Les interrupcions s'habiliten i es desactiven mitjançant el registre INTCON també explicat en l'apartat

anterior.

```
// CONFIGURACIÓ DE LES INTERRUPCIONS
INTCONbits.PEIE = 1; // S'habiliten les interrupcions dels perifèrics
INTCONbits.GIE  = 1;  // S'habiliten les interrupcions globals
INTCONbits.T0IE = 0; // Es desactiven les interrupcions del Timer0
```

Figura 8.23. Habilitació de les interrupcions

A diferència del circuit 1, en aquest circuit no s'ha de configurar l'UART, ja que les dades no les intercanvia amb el mòbil sinó que només les envia cap al dispositiu LCD. Per tant, a continuació, es programa el procediment de la LCD comptant amb la llibreria de funcions corresponent, començant per iniciar el dispositiu i posant en blanc la pantalla, amb les funcions *LCDInit(LS_NONE)* i *LCDClear()*, respectivament (F*igura 8.24*). Posteriorment, es programa perquè es mostri un text per la pantalla, *Figura 8.25*, "Color en RGB" i "TFG". Això és possible gràcies a les funcions *LCDGotoXY(X,Y)*, amb la qual es determina a partir de quina posició X,Y del dispositiu es vol escriure, i *LCDWriteString(string)*, que indica quin text, és a dir, *string*, es vol mostrar.

```
// FUNCIÓ PER INICIALITZAR EL MÒDUL LCD
LCDInit(LS_NONE); // S'inicialitza la LCD
__delay_ms(100); // S'afegeixen delays perquè si es va molt ràpid no funciona, en aquest cas, s'espera 100 milisegons

// FUNCIÓ CLEAR THE DISPLAY
LCDClear(); // La pantalla del dispositiu LCD es posa en blanc
__delay_ms(100); // S'espera 100 milisegons
```

Figura 8.24. Inicialització i posada en blanc del dispositiu LCD

```
// FUNCIÓ WRITE A STRING
LCDGotoXY(2,0); // S'escriu a la pantalla a partir de la posició (2,0), tercera posició de la fila de dalt
__delay_ms(100); // S'espera 100 milisegons
LCDWriteString("Colors en RGB"); // Per la pantalla es mostra el text Colors en RGB
__delay_ms(100); // S'espera 100 milisegons
LCDGotoXY(7,1); // S'escriu a la pantalla a partir de la posició (7,1), vuitena posició de la fila de sota
__delay_ms(100); // S'espera 100 milisegons
LCDWriteString("TFG"); // Per la pantalla es mostra el text TFG
__delay_ms(3000); // S'espera 3 segons
```

Figura 8.25. Procediment per mostrar un text per la pantalla LCD

Després de mostrar el text, cal inicialitzar la variable FREQ, que és com li anomenem a la freqüència captada pel sensor, i configurar que el sensor es connecti de tal manera que l'escala de la freqüència de sortida d'aquest sigui d'un 2%. A més, la pantalla es posa en blanc i s'inicialitzen les variables LEDR, LEDG i LEDB, és a dir, que les components del LED RGB, en definitiva, el LED RGB, començarà apagat. (*Figura 8.26*)

```
FREQ = 0; // S'inicialitza la variable FREQ a 0

// Els pins S0 i S1 del sensor de color es connecten
// de tal manera que l'escala de la seva freqüència de sortida sigui del 2%
S0 = 0;
S1 = 1;

LCDClear(); // Es neteja la pantalla, posant-la en blanc

LEDR = 0;
LEDG = 0;
LEDB = 0;
```

Figura 8.26. Inicialització de la variable FREQ, configuració dels pins del sensor i posada en blanc de la pantalla. També, inicialització de les variables LEDR, LEDG i LEDB

Posteriorment, comença el bucle de la funció *main*. Es realitza un escombrat pel qual el sensor primer es connecta per mesurar la freqüència amb el filtre vermell, després verd, blau i, finalment, el filtre blanc. Abans de canviar de filtre, la variable de la FREQ de cada color, és a dir, pel filtre vermell serà FREQ_R, pren el valor de la funció que mesura la freqüència. Aquest valor, es multiplica per 4 perquè a la *funció mesura_frequencia*, com es veurà a continuació, s'agafa el nombre d'oscil·lacions que hi ha en un quart de segon. A més, el valor de la freqüència s'ha multiplicat pel factor 255/1316 per convertir-lo en un valor entre el 0 i el 255, seguint el procediment explicat a l'apartat 5.2. La variable Valor_R pren el valor d'aquest nombre entre el 0 i el 255. Seguidament, a la pantalla s'escriu la inicial del color en anglès seguida de dos punts, és a dir que en el cas del vermell es mostraria "R:", amb la funció LCDWriteString("R:"), en el cas del verd es mostraria "G:", pel blau, "B:" i pel blanc, "C:". Aquest text es comença a escriure a la posició indicada per la funció LCDGotoXY(X,Y), on la X indica la fila de la pantalla i la Y, la columna. Seguint amb el vermell, la variable Valor_R es mostra per pantalla després de "R:" amb la funció LCDWriteInt(Valor_R, 5), on el 5 indica el nombre de xifres que es volen mostrar. A continuació, a la *Figura 8.27*, es mostra la programació el bucle, primer es realitza el procediment pel vermell, després pel verd, seguint amb el blau i s'acaba amb el blanc. Tot aquest escombrat es realitza sempre i quan el polsador no estigui premut, la variable boto serà igual a no_premut, és a dir, 1 i, a més, la variable estat valgui 0, ja que abans de començar a realitzar el procediment hi ha un *if*.

```
while (1) // Mentre el contingut del parèntesi sigui 1
{
    if (boto==no_premut && estat== 0) // Quan la variable boto sigui 1 i estat sigui 0, hi haurà el següent procediment
    {
        // RED - VERMELL
        S2 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre vermell
        S3 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre vermell
        FREQ_R = mesura_frequencia(); // La variable FREQ_R pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(0,0); // S'escriu a la pantalla a partir de la posició (0,0), primera posició de la fila de dalt
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("R:"); // Mostrar el text R:
        Valor_R=(4*FREQ_R)*(255/1316.); // La variable Valor_R pren el valor de la freqüència en vermell
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_R es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_R,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres

        // GREEN - VERD
        S2 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre verd
        S3 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre verd
        FREQ_G = mesura_frequencia(); // La variable FREQ_G pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(8,0); // S'escriu a la pantalla a partir de la posició (8,0), novena posició de la fila de dalt
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("G:"); // Mostrar el text G:
        Valor_G=(4*FREQ_G)*(255/1316.); // La variable Valor_G pren el valor de la freqüència en verd
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_G es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_G,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres

        // BLUE - BLAU
        S2 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blau
        S3 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blau
        FREQ_B = mesura_frequencia(); // La variable FREQ_B pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(0,1); // S'escriu a la pantalla a partir de la posició (0,1), primera posició de la fila de sota
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("B:"); // Mostrar el text B:
        Valor_B=(4*FREQ_B)*(255/1316.); // La variable Valor_B pren el valor de la freqüència en blau
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_B es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_B,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres

        // CLEAR - BLANC
        S2 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blanc
        S3 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blanc
        FREQ_C = mesura_frequencia(); // La variable FREQ_C pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(8,1); // S'escriu a la pantalla a partir de la posició (8,1), novena posició de la fila de sota
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("C:"); // Mostrar el text C:
        Valor_C=(4*FREQ_C)*(255/1316.); // La variable Valor_C pren el valor de la freqüència en blanc
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_C es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_C,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres
```

Figura 8.27. Escombrat de la freqüència amb els diferents filtres: vermell, verd, blau i blanc

Després de la realització de l'escombrat de les freqüències, es vol que el valor entre 0 i 255 que s'havia mostrat per la pantalla sigui un número entre el 0 i el 100. S'aconsegueix multiplicant per 100 el valor que capta el sensor per cada filtre, és a dir, la variable FREQ_R, FREQ_G o FREQ_C, i dividint-lo per la suma d'aquests tres valors. Aquests números entre el 0 i el 100 són els que prenen les variables amplepolsR, amplepolsG i amplepolsB, *Figura 8.28*. D'aquesta manera es troba el percentatge que hi ha de cada color, la qual cosa és útil per a la funció que serveix per encendre el LED RGB.

```
        amplepolsR = (FREQ_R*100)/(FREQ_R+FREQ_G+FREQ_B); // Es passa el valor de FREQ_R entre un numero del 0 al 100
        amplepolsG = (FREQ_G*100)/(FREQ_R+FREQ_G+FREQ_B); // Es passa el valor de FREQ_G entre un numero del 0 al 100
        amplepolsB = (FREQ_B*100)/(FREQ_R+FREQ_G+FREQ_B); // Es passa el valor de FREQ_B entre un numero del 0 al 100

    } // Final de la condició de l'if
```

Figura 8.28. Variables amplepolsR, amplepolsG i amplepolsB

A continuació, s'inicia el procediment per encendre el LED RGB mitjançant el polsador. Primer es crea una condició per la qual si el polsador no està premut i la variable estat és 1, el programa principal no fa res, per a què aquest es quedi amb l'última lectura del sensor de color. En canvi, quan el polsador està premut, la variable boto és igual a premut, és a dir, quan està a 0, es desactiven les interrupcions del *Timer0*, l'estat canvia el seu valor i les components del LED RGB s'apaguen. A més, mentre el polsador està premut i si la variable estat és 1, s'habiliten les interrupcions del *Timer0*, mitjançant les quals s'encén el LED RGB segons la llum que es necessita de cada component del LED, la vermella, la verda i la blava. Això es deu a que les interrupcions del *Timer0* s'han utilitzat per crear una funció que té la mateixa funcionalitat que el mòdul PWM del microcontrolador. La informació sobre la funcionalitat PWM s'explica detalladament en el capítol anterior. Després d'aquesta part del programa finalitza la funció principal, *Figura 8.29*.

```
        //INICI DEL PROCEDIMENT PER ENCENDRE EL LED RGB
        if (boto==no_premut && estat==1) // Quan la variable boto sigui igual a no_premut i estat sigui 1
        {
        // No es realitza cap càlcul per a què es quedi amb l'última lectura del sensor de color
        } // Final de la condició de l'if

        // Per controlar quan s'encén el LED RGB
        if (boto==premut) // Quan la variable boto sigui igual a premut
        {
            INTCONbits.T0IE = OFF; // Es desactiven les interrupcions del Timer0
            estat=!estat; // L'estat canviarà el seu valor, passarà de 0 a 1 o, en el cas en què fos 1, passaria a ser 0
            LEDR = 0; // El LED vermell del RGB s'apaga
            LEDG = 0; // El LED verd del RGB s'apaga
            LEDB = 0; // El LED blau del RGB s'apaga
            while (boto==premut); // El programa espera en aquest punt mentre no s'allibera el polsador
            if (estat == 1)
            {
                INTCONbits.T0IE = ON; // S'habiliten les interrupcions del Timer0
            } // Final de la condició de l'if
            __delay_ms(100); // S'espera 100 milisegons. S'afegeix el delay per evitar rebots del polsador
        } // Final de la condició de l'if


    } // Final del bucle while
} // Final de la funció principal
```

Figura 8.29. Procediment on intervé el polsador per encendre el LED RGB

Una vegada finalitzada la funció *main*, es programa la funció *mesura_frequencia* que, com s'ha comentat anteriorment, és la que capta la informació del sensor de color, *Figura 8.30*. Això ho pot fer gràcies al *Timer1*. Aquesta funció és comuna pels dos circuits així que s'explica a l'apartat anterior.

```
// FUNCIÓ QUE MESURA LA FREQÜÈNCIA R, G, B I CLEAR
unsigned int mesura_frequencia(void) //Funció mesura_frequencia, serveix per captar la informació del sensor
    {
    T1CONbits.TMR1ON = 1; // S'activa el Timer1
    TMR1 = 0; // El Timer1 es posa a 0
    __delay_ms(250); // S'espera un quart de segon
    FREQ = TMR1; // Es llegeix el valor del Timer1 i aquest valor el pren la variable FREQ
    T1CONbits.TMR1ON = 0; // Es desactiva el Timer1
    return(FREQ); // Retorna el valor de la variable FREQ, que és el valor del Timer1
    }
```

Figura 8.30. Funció mesura_frequencia

Per acabar amb la programació, es programa el servei d'interrupcions, *Figura 8.31*, on el programa entra cada vegada que hi ha una interrupció. Un cop s'entra en aquesta rutina, hi ha una condició amb un *if*. Com en el circuit 1, la funció d'aquest *if* és modular l'amplitud del pols de l'ona de cada filtre (vermell, verd i blau). Quan s'activa la bandera d'interrupció del Timer0, és a dir, el bit T0IF del registre INTCON es posa a 1, els comptadors compteR, compteG i compteB incrementen una unitat. Quan el comptador arriba a l'amplitud del pols més 1, la component del LED RGB s'apaga. D'aquesta manera, es modula el temps que està encesa cada component. A més, com passa en el circuit 1, se li suma una unitat a l'amplitud del pols perquè, com que cada vegada que s'entra al servei d'interrupcions el comptador incrementa, si l'amplitud del pols fos 0, el LED mai s'apagaria. Quan la variable compteR arriba a 100, s'agafa aquesta variable però tots els comptadors són iguals en tot moment, les tres components del LED s'encenen i els comptadors es posen a 0. Finalment, el Timer0 s'inicialitza a 215, si fos un valor més baix es veurien pampallugues en el LED, i es desactiva la bandera d'interrupció del *Timer0*.

```
// SERVEI D'INTERRUPCIONS PER PODER ENCENDRE EL LED RGB SEGONS EL COLOR CAPTAT PEL SENSOR
void interrupt Interrupcion()   // Funció que atén les interrupcions
{
    if (INTCONbits.T0IF) // Sempre que la bandera d'interrupció del Timer0 estigui activada
    {
        compteR++; // La variable compteR augmenta una unitat cada vegada que hi ha una interrupció
        compteG++; // La variable compteG augmenta una unitat cada vegada que hi ha una interrupció
        compteB++; // La variable compteB augmenta una unitat cada vegada que hi ha una interrupció
        if (compteR == amplepolsR + 1) // Quan la variable compteR és igual a amplepolsR més 1
                                       // (si no se li suma 1 a amplepolsR, aquesta mai podria ser 0)
        {
            LEDR = 0; // La component vermella del LED RGB s'apaga
        }
        if (compteG == amplepolsG + 1) // Quan la variable compteG és igual a amplepolsG més 1
                                       // (si no se li suma 1 a amplepolG, aquesta mai podria ser 0)
        {
            LEDG = 0; // La component verda del LED RGB s'apaga
        }
        if (compteB == amplepolsB + 1) // Quan la variable compteB sigui igual a amplepolsB més 1
                                       // (si no se li suma 1 a amplepolsB, aquesta mai podria ser 0)
        {
            LEDB = 0; // La component blava del LED RGB s'apaga
        }
        if (compteR == 100) // Quan el compteR val 100 es segueix amb el procediment següent
                            // (les variables compteR, compteG i compteB són iguals en tot moment)
        {
            compteR = 0; // La variable compteR torna a 0
            compteG = 0; // La variable compteG torna a 0
            compteB = 0; // La variable compteB torna a 0
            LEDR = 1; // La compoment vermella del LED RGB s'encén
            LEDG = 1; // La compoment verda del LED RGB s'encén
            LEDB = 1; // La compoment blava del LED RGB s'encén
        }
        TMR0 = 215; // El Timer0 es posa a 215, un valor prou alt perquè no es vegin pampallugues al LED RGB
        INTCONbits.T0IF = 0;  // Es desactiva la bandera d'interrupció del Timer0
    }
}
```

Figura 8.31. Servei d'interrupcions del programa

Per últim, cal comentar que durant el programa hi ha una funció anomenada *__delay_ms*, el que fa aquesta funció és provocar un retard, ja que si el programa anés massa ràpid, podria ser que no funcionés bé.

9. Pressupost

En aquest apartat es realitza un pressupost econòmic total del projecte. Els costos es desglossen en costos materials, costos de llicències, costos d'equipaments i costos de personal.

COST TOTAL PROJECTE					
COSTOS MATERIALS					
Concepte	Quantitat	Preu unitari (€)			Preu total (€)
Protoboard Ariston	2	18,55			37,10
Microcontrolador PIC16F690	2	2,70			5,40
Sensor TCS3210	2	2,18			4,36
Programador PICkit2	2	24,99			49,98
Mòdul bluetooth HC-05	1	4,90			4,90
Cables	60	0,03			1,80
LED	6	0,18			1,08
LED RGB	2	1,30			2,60
Resistència	6	1,19			7,14
Pila 9V	2	1,30			2,60
Bobina estany	1	2,70			2,70
Pins connectors	70	0,02			1,40
Connector pila 9V	2	0,42			0,84
Placa perforada	2	1,24			2,48
Soldador Cetronic	1	7.52			7.52
Pantalla LCD	1	3,05			3,05
Polsador	1	0,21			0,21
		Cost total material (€)			**127,64**
COSTOS LLICÈNCIES					
Concepte		Preu llicència (€)			Preu total (€)
MPLAB IDE		0			0,00
Compilador XC8		0			0,00
MIT App Inventor 2		0			0,00
		Cost total llicències (€)			**0**
COSTOS EQUIPAMENT					
Concepte	Quantitat	Preu (€)	Amortització (anys)	Ús (anys)	Preu total (€)
Ordinador ACER TravelMate 8471	1	630,00	5	0,5	63,00
Ordinador ASUS VivoBook X541UA	1	681,68	5	0,5	68,17
Smartphone BQ U Plus	1	200,00	2	0,5	50,00
			Cost total material (€)		**181,17**

COSTOS DE PERSONAL			
Concepte	Preu/hora (€/h)	Hores (h)	Preu total (€)
Estudi previ	15	40	600
Proves	15	40	600
Muntatge	15	30	450
Programació	15	100	1500
Redacció memòria	15	130	1950
Dedicació tutora	50	30	1500
	Cost total de personal (€)		**4950**
COST TOTAL DEL PROJECTE			**5258,81 €**

10. Impacte ambiental

Els aparells electrònics tenen un major impacte ambiental durant el seu ús quotidià i al final de la seva vida útil.

Durant l'etapa d'ús del projecte, l'impacte que produeix és a causa de les ones de radiofreqüència que hi ha durant la comunicació entre el *Bluetooth* i el mòbil, ja que els dos dispositius generen aquestes ones. Les ones de radiofreqüència s'enllacen en la banda ISM (de l'anglès *Industrial, Scientific, Medical*), banda reservada per l'ús no comercial de radiofreqüència electromagnètica en àrees industrial, científica i mèdica i popularitzades pel seu ús en comunicacions. L'ús d'aquestes bandes està obert a tothom sense la necessitat de tenir una llicència, sempre i quan es respectin les regulacions que limiten els nivells de potència transmesa, ja que el fet de sobrepassar-los podria provocar danys. Com que el projecte és senzill, només utilitza un dispositiu mòbil i un *Bluetooth*, aquest tipus d'impacte ambiental s'ha considerat insignificant en relació a la contaminació que pot causar un aparell electrònic quan se li acaba el seu temps de funcionament, ja que els residus elèctrics necessiten tractar-se amb especial cura.

A mesura que avancen els anys i milloren les tecnologies, els aparells electrònics són cada vegada més abundants, el que fa que els seus residus augmentin de manera exponencial. A més, com que la varietat en els aparells és tan gran, això fa que, enlloc d'intentar reparar-los vagin directament a les escombraries quan, normalment es tracta de residus tòxics, com per exemple el brom, el mercuri o el fòsfor, nocius per la salut i pel medi ambient.

Afortunadament, la conscienciació de les persones amb el medi ambient va sent més elevada i les legislacions ambientals són més estrictes i exigents, per tant, ajuda a que l'impacte ambiental dels aparells elèctrics i electrònics sigui menor.

Per altra banda, hi ha aparells que es poden reciclar per peces i, a més, existeixen punts de recollida on els usuaris poden dur els aparells dels quals se'n volen desfer i allà els tracten de manera especial, separant les substàncies tòxics dels altres components, com els plàstics o l'alumini, els quals, molts d'ells, es poden reciclar, posteriorment, per fer noves peces.

Conclusions

L'objectiu principal d'aquest projecte era dissenyar i construir aparells electrònics capaços d'identificar un color i poder reproduir-lo. Treballar en equip ha permès que s'hagi pogut arribar a l'objectiu per diferents vies, fet que enriqueix molt un projecte.

Abans de començar qualsevol tipus de projecte cal conèixer bé les eines de treball que s'utilitzaran per tal d'entendre els resultats obtinguts. És per això que s'ha hagut, per exemple, de caracteritzar el sensor o provar el funcionament de la pantalla LCD abans de començar a dissenyar el sistema definitiu. Al realitzar un projecte d'aquesta envergadura, s'ha après que és important treballar poc a poc i entenent cada procediment que es fa. És molt més fàcil detectar un error si es va provant per parts i, a més, s'acaba coneixent millor el sistema.

Les autores han quedat satisfetes de poder complir amb les especificacions establertes a l'inici del projecte. S'ha pogut aprofundir en alguns àmbits de l'electrònica d'una manera més amplia, aprenent a treballar amb components, fins al moment desconeguts, i arribant amb ells als objectius proposats. Per una banda, s'ha aconseguit captar un feix de llum, analitzar la freqüència de les seves ones i mostrar el valor dels components del color RGB per una pantalla LCD. A més, s'ha reproduït aquest color per un LED RGB. Per altra banda, mitjançant un telèfon mòbil i un mòdul *Bluetooth*, s'ha aconseguit fer el mateix que en el cas de la pantalla LCD però mostrant els resultats per la pantalla del mòbil. A part, amb el mòbil també s'ha aconseguit que l'usuari pugui generar un color i reproduir-lo amb l'ajuda d'un LED RGB. D'aquesta manera, ambdues integrants del grup han treballat amb components comuns, com ara el sensor o el LED RGB, però han utilitzat diferents alternatives a l'hora de mostrar els resultats obtinguts.

Durant la realització d'aquest treball hi ha hagut problemes i contratemps que s'han hagut d'anar solucionant buscant estratègia per tal d'acotar l'origen d'aquests. Les autores d'aquest treball, tot i tenir circuits i programes diferents, s'han anat ajudant entre elles per tal de superar aquests entrebancs.

Tot i haver complert amb les expectatives del projecte, hi ha alguns punts que es podrien millorar essent el més clar la baixa qualitat del sensor de color utilitzat. Per exemple, en algunes ocasions, el color reproduït difereix del color captat pel sensor i la raó d'aquesta discrepància es troba en la resposta del sensor.

Agraïments

Tot el nostre agraïment a la nostra professora i tutora Rosa Rodríguez, del departament d'electrònica de l'ETSEIB, per tota l'ajuda que ens ha donat al llarg d'aquests últims mesos, dedicant-nos el seu temps per poder dur a terme aquest projecte. Ens ha guiat en tot moment per encaminar el treball correctament i ha estat disposada a solucionar qualsevol contratemps.

També ens agradaria agrair al Departament d'Electrònica de l'Escola que ens ha posat a la nostra disposició el laboratori i ens ha proporcionat el material necessari per a implementar els sistemes electrònics d'aquest projecte.

Finalment, volem donar les gràcies a les respectives famílies, ja que sense el seu suport i encoratjament no hagués sigut possible arribar fins aquí.

Bibliografia

[1] Microchip. (2016). *PIC16F690 Datasheet*. [online]
Disponible a: http://ww1.microchip.com/downloads/en/DeviceDoc/41262E.pdf

[2] Microchip. (2016). *PICkit 2 User's Guide*. [online]
Disponible a: http://ww1.microchip.com/downloads/en/DeviceDoc/51553E.pdf

[3] Microchip Technology Inc. *MPLAB XC8 C COMPILER User's Guide*. 2012.

[4] TAOS Inc. (2009) *TCS3200, TCS3210 Programmable color Light-to-frequency converter Datasheet*

[5] ITead Studio. (2016). *HC-05 Datasheet*. [online]
Disponible a: http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf

[6] HITACHI. (2016). *HD44780U LCD Datasheet* [online]
Disponible a: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

[7] KINGBRIGHT (2010). *L-154A4SURKQBDZGW Datasheet* [online]
Disponible a: http://www.ges.cz/sheets/l/l154a4surkqbdzgw.pdf

[8] Ai2.appinventor.mit.edu. (2016). *MIT App Inventor 2*. [online]
Disponible en: http://ai2.appinventor.mit.edu/